Red Hat Enterprise Linux 9

파일 시스템 관리

Red Hat Enterprise Linux 9에서 파일 시스템 생성, 수정 및 관리

Last Updated: 2024-07-19

Red Hat Enterprise Linux 9 파일 시스템 관리

Red Hat Enterprise Linux 9에서 파일 시스템 생성, 수정 및 관리

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 문서 컬렉션은 Red Hat Enterprise Linux 9에서 파일 시스템을 효과적으로 관리하는 방법에 대한 지침을 제공합니다.

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

문서에 대한 피드백에 감사드립니다. 어떻게 개선할 수 있는지 알려주십시오.

특정 문구에 대한 의견 제출

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

Bugzilla를 통해 피드백 제출(등록 필요)

1. Bugzilla 웹 사이트에 로그인합니다.
2. **버전** 메뉴에서 올바른 버전을 선택합니다.
3. **Summary** (요약) 필드에 설명 제목을 입력합니다.
4. **Description** (설명) 필드에 개선을 위한 제안을 입력합니다. 문서의 관련 부분에 대한 링크를 포함합니다.
5. **버그 제출**을 클릭합니다.

1장. 사용 가능한 파일 시스템 개요

애플리케이션에 적합한 파일 시스템을 선택하는 것은 사용 가능한 많은 옵션과 관련된 장단점으로 인해 중요한 결정입니다. 이 장에서는 Red Hat Enterprise Linux 9와 함께 제공되는 일부 파일 시스템에 대해 설명하고 애플리케이션에 맞는 올바른 파일 시스템에 대한 기록 배경과 권장 사항을 제공합니다.

1.1. 파일 시스템 유형

Red Hat Enterprise Linux 9는 다양한 파일 시스템(FS)을 지원합니다. 다양한 유형의 파일 시스템을 통해 다양한 종류의 문제를 해결하고 용도는 애플리케이션에 따라 다릅니다. 가장 일반적인 수준에서 사용 가능한 파일 시스템을 다음과 같은 주요 유형으로 그룹화할 수 있습니다.

표 1.1. 파일 시스템 유형 및 사용 사례

유형	파일 시스템	속성 및 사용 사례
디스크 또는 로컬 FS	XFS	XFS는 RHEL의 기본 파일 시스템입니다. Extent로 파일을 배치하기 때문에 ext4보다 조각화에 덜 취약합니다. 별도로 수행해야 할 이유가 없는 한 로컬 파일 시스템으로 XFS를 로컬 파일 시스템으로 배포할 것을 권장합니다(예: 성능 관련 호환성 또는 코너 케이스).
	ext4	ext4는 Linux에서 장기간의 이점을 제공합니다. 따라서 거의 모든 Linux 애플리케이션에서 지원됩니다. 대부분의 경우 성능이 XFS입니다. ext4는 일반적으로 홈 디렉터리에 사용됩니다.
네트워크 또는 클라이언트-서버 FS	NFS	NFS를 사용하여 동일한 네트워크의 여러 시스템 간에 파일을 공유합니다.
	SMB	Microsoft Windows 시스템과 파일 공유를 위해 SMB를 사용합니다.
공유 스토리지 또는 공유 디스크 FS	GFS2	GFS2에서는 컴퓨팅 클러스터의 멤버에 공유 쓰기 액세스 권한을 제공합니다. 안정성과 신뢰성에 중점을 두고 있으며 가능한 한 로컬 파일 시스템의 기능적 경험을 갖추고 있습니다. SAS Grid, Tibcolibvirt, IBM Websphere 6443 및 Red Hat Active950이 성공적으로 배포되어 있습니다.
볼륨 관리 FS	Stratis(기술 프리뷰)	Stratis는 XFS 및 LVM의 조합을 기반으로 구축된 볼륨 관리자입니다. Stratis의 목적은 RuntimeClass 및 RuntimeClass와 같은 볼륨 관리 파일 시스템에서 제공하는 기능을 에뮬레이션하는 것입니다. 이 스택을 수동으로 빌드할 수 있지만 Stratis는 구성 복잡성을 줄이고, 모범 사례를 구현하고, 오류 정보를 통합합니다.

1.2. 로컬 파일 시스템

로컬 파일 시스템은 단일 로컬 서버에서 실행되며 스토리지에 직접 연결된 파일 시스템입니다.

예를 들어 로컬 파일 시스템은 내부 SATA 또는 SAS 디스크에 대해 유일한 선택이며 서버에 로컬 드라이브가 있는 하드웨어 RAID 컨트롤러가 있는 경우 사용됩니다. SAN에서 내보낸 장치가 공유되지 않은 경우 로컬 파일 시스템도 SAN 연결 스토리지에서 사용되는 가장 일반적인 파일 시스템입니다.

모든 로컬 파일 시스템은 POSIX와 호환되며 지원되는 모든 Red Hat Enterprise Linux 릴리스와 완벽하게 호환됩니다. POSIX 호환 파일 시스템은 **read()**, **write()** 및 search()와 같이 잘 정의된 시스템 호출 세트를 지원합니다.

애플리케이션 프로그래밍 프로그램의 관점에서 볼 때 로컬 파일 시스템간에 상대적으로 약간의 차이점이 있습니다. 사용자 관점과 가장 주목할 만한 차이점은 확장성 및 성능과 관련이 있습니다. 파일 시스템 선택을 고려할 때 파일 시스템의 크기, 보유 가능한 고유한 기능 및 워크로드에서 수행하는 방법을 고려하십시오.

사용 가능한 로컬 파일 시스템

- **XFS**
- **ext4**

1.3. XFS 파일 시스템

XFS는 단일 호스트에서 매우 큰 파일 및 파일 시스템을 지원하는 확장성이 뛰어난 고성능, 견고하며 성숙한 64비트 저널링 파일 시스템입니다. Red Hat Enterprise Linux 9의 기본 파일 시스템입니다. XFS는 원래 SGI에 의해 Subversions에서 개발되었으며 매우 큰 서버 및 스토리지 어레이에서 실행하는 오랜 기록을 보유하고 있습니다.

XFS의 기능은 다음과 같습니다.

신뢰성

- **메타데이터 저널링 - 시스템을 다시 시작하고 파일 시스템을 다시 시작할 때 다시 마운트할 수 있는 파일 시스템 작업 레코드를 유지하여 시스템 충돌 후 파일 시스템 무결성을 보장합니다.**
- **광범위한 런타임 메타데이터 일관성 확인**
- **확장 가능하고 빠른 복구 유틸리티**

- 할당량 저널링. 이렇게 하면 충돌 후 긴 할당량 일관성 점검이 필요하지 않습니다.

확장 및 성능

- 지원되는 파일 시스템 크기는 최대 **1024TiB**
- 다수의 동시 작업을 지원하는 기능
- 여유 공간 관리의 확장성을 위한 **B-tree** 인덱싱
- 정교한 메타데이터 읽기 알고리즘
- 비디오 워크로드 스트리밍 최적화

할당 체계

- 범위 기반 할당
- 스트라이프 인식 할당 정책
- 지연된 할당
- 공간 사전 할당
- 동적으로 할당된 **inode**

기타 기능

- **reflink** 기반 파일 복사
- 긴밀하게 통합된 백업 및 복원 유틸리티

- 온라인 조각 모음
- 온라인 파일 시스템 확장
- 포괄적인 진단 기능
- 확장 속성(xattr). 이를 통해 시스템은 파일당 여러 개의 추가 이름/값 쌍을 연결할 수 있습니다.
- 프로젝트 또는 디렉터리 할당량입니다. 이렇게 하면 디렉터리 트리에 대한 할당량 제한을 사용할 수 있습니다.
- 2초 타임 스탬프

성능 특성

XFS는 엔터프라이즈 워크로드가 있는 대규모 시스템에서 고성능을 보유하고 있습니다. 대규모 시스템은 비교적 많은 CPU, 여러 HBA 및 외부 디스크 어레이 연결을 사용하는 시스템입니다. XFS는 멀티 스레드 병렬 I/O 워크로드가 있는 소규모 시스템에서도 제대로 작동합니다.

XFS는 단일 스레드에 대해 상대적으로 낮은 성능을 제공합니다. 예를 들어 단일 스레드에서 많은 수의 작은 파일을 생성하거나 삭제하는 워크로드와 같습니다.

1.4. EXT4 파일 시스템

ext4 파일 시스템은 ext 파일 시스템 제품군의 네 번째 세대입니다. Red Hat Enterprise Linux 6의 기본 파일 시스템입니다.

ext4 드라이버는 ext2 및 ext3 파일 시스템을 읽고 쓸 수 있지만 ext4 파일 시스템 형식은 ext2 및 ext3 드라이버와 호환되지 않습니다.

ext4는 다음과 같은 몇 가지 새로운 기능과 향상된 기능을 추가합니다.

- 지원되는 파일 시스템 크기 최대 **50TiB**
- 확장 기반 메타데이터
- 지연된 할당
- **journal checksumming**
- 대규모 스토리지 지원

범위 기반 메타데이터 및 지연된 할당 기능은 파일 시스템에서 사용된 공간을 추적하는 더 작고 효율적인 방법을 제공합니다. 이러한 기능을 통해 파일 시스템 성능을 개선하고 메타데이터로 소비되는 공간을 줄일 수 있습니다. 지연 할당을 사용하면 데이터가 디스크로 플러시될 때까지 새로 기록된 사용자 데이터에 대한 영구 위치를 파일 시스템을 연기할 수 있습니다. 이를 통해 더 크고 연속된 할당을 허용할 수 있으므로 파일 시스템이 훨씬 더 나은 정보로 결정을 내릴 수 있으므로 성능이 향상됩니다.

ext4에서 **fsck** 유틸리티를 사용하는 파일 시스템 복구 시간은 **ext2** 및 **ext3**보다 훨씬 빠릅니다. 일부 파일 시스템 복구는 성능이 **6**배 증가했습니다.

1.5. XFS 및 EXT4 비교

XFS는 **RHEL**의 기본 파일 시스템입니다. 이 섹션에서는 **XFS** 및 **ext4**의 사용과 기능을 비교합니다.

메타데이터 오류 동작

ext4에서는 파일 시스템에 메타데이터 오류가 발생할 때 동작을 구성할 수 있습니다. 기본 동작은 작업을 계속하는 것입니다. **XFS**에서 복구할 수 없는 메타데이터 오류가 발생하면 파일 시스템을 종료하고 **EFSCORRUPTED** 오류를 반환합니다.

할당량

ext4에서는 기존 파일 시스템에서 파일 시스템을 만들 때 할당량을 활성화할 수 있습니다. 그런 다음 마운트 옵션을 사용하여 할당량 적용을 구성할 수 있습니다.

XFS 할당량은 다시 마운트할 수 없는 옵션이 아닙니다. 초기 마운트에서 할당량을 활성화해야 합니다.

XFS 파일 시스템에서 **quotacheck** 명령을 실행하면 영향을 미치지 않습니다. 할당량 계정을 처음 켜면 **XFS**는 할당량을 자동으로 확인합니다.

파일 시스템 크기 조정

XFS는 파일 시스템의 크기를 줄이는 유틸리티가 없습니다. **XFS** 파일 시스템의 크기만 늘릴 수 있습니다. 반면 **ext4**는 파일 시스템의 크기 확장 및 축소를 지원합니다.

inode 번호

ext4 파일 시스템은 2^{32} 개의 **inode**를 지원하지 않습니다.

XFS는 동적으로 **inode**를 할당합니다. 파일 시스템에 여유 공간이 있는 한 **XFS** 파일 시스템은 **inode**에서 실행되지 않습니다.

특정 애플리케이션은 **XFS** 파일 시스템에서 2^{32} 보다 큰 **inode** 번호를 올바르게 처리할 수 없습니다. 이러한 애플리케이션은 **EOVERFLOW** 반환 값을 사용하여 **32**비트 **stat** 호출을 실패할 수 있습니다. **inode** 번호는 다음 조건에서 2^{32} 를 초과합니다.

- 파일 시스템은 **256**바이트 **inode**가 있는 **1TiB**보다 큽니다.
- 파일 시스템은 **512**바이트 **inode**가 있는 **2TiB**보다 큽니다.

애플리케이션이 큰 **inode** 번호로 실패하면 **-o inode**^{32} 옵션을 사용하여 **XFS** 파일 시스템을 마운트하여 **232** 미만의 **inode** 번호를 적용합니다. **inode32** 를 사용하면 **64**비트 번호로 이미 할당된 **inode**에는 영향을 미치지 않습니다.

중요

특정 환경에 필요한 경우가 아니면 **inode32** 옵션을 *사용하지* 마십시오. **inode32** 옵션은 할당 동작을 변경합니다. 결과적으로 더 낮은 디스크 블록에서 **inode**를 할당하는 데 사용할 수 있는 공간이 없는 경우 **ENOSPC** 오류가 발생할 수 있습니다.

1.6. 로컬 파일 시스템 선택

애플리케이션 요구 사항을 충족하는 파일 시스템을 선택하려면 파일 시스템을 배포할 대상 시스템을

이해해야 합니다. 다음 질문을 사용하여 결정을 알릴 수 있습니다.

- 대규모 서버가 있습니까?
- 대규모 스토리지 요구 사항이 있거나 로컬의 느린 SATA 드라이브가 있습니까?
- 애플리케이션이 제공될 것으로 예상되는 I/O 워크로드의 유형은 무엇입니까?
- 처리량 및 대기 시간 요구 사항은 무엇입니까?
- 서버 및 스토리지 하드웨어의 안정성은 무엇입니까?
- 파일 및 데이터 세트의 일반적인 크기는 무엇입니까?
- 시스템에 오류가 발생하면 다운타임이 발생할 수 있습니까?

서버와 스토리지 장치가 모두 큰 경우 XFS를 선택하는 것이 좋습니다. 크기가 작은 스토리지 어레이의 경우에도 XFS는 평균 파일 크기가 클 때 매우 잘 작동합니다(예: 크기가 수백 메가바이트).

기존 워크로드가 ext4와 함께 잘 수행된 경우 ext4를 계속 사용하면 애플리케이션에 매우 친숙한 환경을 제공해야 합니다.

ext4 파일 시스템은 I/O 기능이 제한된 시스템에서 더 잘 작동하는 경향이 있습니다. 제한된 대역폭(200MB/s 미만)과 최대 1000 IOPS 기능에서 더 잘 작동합니다. 높은 기능을 가진 모든 것에 대해 XFS는 더 빠른 경향이 있습니다.

XFS는 ext4와 비교하여 CPU당 두 배 작업을 사용하므로 CPU 바인딩 워크로드를 약간의 동시성이 있는 경우 ext4가 더 빠릅니다. 일반적으로 애플리케이션이 단일 읽기/쓰기 스레드와 작은 파일을 사용하는 경우 ext4가 더 나은 반면, 애플리케이션이 여러 읽기/쓰기 스레드와 더 큰 파일을 사용하는 경우 XFS는 축소됩니다.

XFS 파일 시스템을 축소할 수 없습니다. 파일 시스템을 축소할 수 있어야 하는 경우 오프라인 축소를 지원하는 ext4를 사용하는 것이 좋습니다.

일반적으로 **Red Hat**은 **ext4**의 특정 사용 사례가 없는 한 **XFS**를 사용하도록 권장합니다. 또한 적절한 유형의 파일 시스템을 선택하도록 대상 서버 및 스토리지 시스템에서 특정 애플리케이션의 성능을 측정해야 합니다.

표 1.2. 로컬 파일 시스템 권장 사항 요약

시나리오	권장되는 파일 시스템
특별한 사용 사례 없음	XFS
대규모 서버	XFS
대규모 스토리지 장치	XFS
대용량 파일	XFS
다중 스레드 I/O	XFS
단일 스레드 I/O	ext4
제한된 I/O 기능 (1000 IOPS 미만)	ext4
제한된 대역폭 (200MB/s)	ext4
CPU 바인딩된 워크로드	ext4
오프라인 축소 지원	ext4

1.7. 네트워크 파일 시스템

네트워크 파일 시스템(클라이언트/서버 파일 시스템이라고도 함)을 사용하면 클라이언트 시스템이 공유 서버에 저장된 파일에 액세스할 수 있습니다. 따라서 여러 시스템의 여러 사용자가 파일 및 스토리지 리소스를 공유할 수 있습니다.

이러한 파일 시스템은 파일 시스템 집합을 하나 이상의 클라이언트로 내보내는 하나 이상의 서버에서 빌드됩니다. 클라이언트 노드는 기본 블록 스토리지에 액세스할 수 없지만 더 나은 액세스 제어를 허용하는 프로토콜을 사용하여 스토리지와 상호 작용합니다.

사용 가능한 네트워크 파일 시스템

- **RHEL** 고객의 가장 일반적인 클라이언트/서버 파일 시스템은 **NFS** 파일 시스템입니다. **RHEL**은 네트워크를 통해 로컬 파일 시스템을 내보내는 **NFS** 서버 구성 요소와 이러한 파일

시스템을 가져올 **NFS** 클라이언트를 모두 제공합니다.

- **RHEL**에는 **Windows** 상호 운용성을 위한 널리 사용되는 **Microsoft SMB** 파일 서버를 지원하는 **CIFS** 클라이언트도 포함되어 있습니다. 사용자 공간 **Samba** 서버는 **Windows** 클라이언트에 **RHEL** 서버의 **Microsoft SMB** 서비스를 제공합니다.

1.8. 공유 스토리지 파일 시스템

클러스터 파일 시스템(클러스터 파일 시스템)이라고도 하는 공유 스토리지 파일 시스템은 클러스터의 각 서버에 로컬 **SAN(Storage Area Network)**을 통해 공유 블록 장치에 대한 직접 액세스 권한을 제공합니다.

네트워크 파일 시스템과 비교

클라이언트/서버 파일 시스템과 마찬가지로 공유 스토리지 파일 시스템은 클러스터의 모든 멤버인 서버 집합에서 작동합니다. 그러나 **NFS**와 달리 단일 서버는 다른 멤버에게 데이터 또는 메타데이터에 대한 액세스를 제공하지 않습니다. 클러스터의 각 멤버는 동일한 스토리지 장치(*공유*스토리지)에 직접 액세스할 수 있으며 모든 클러스터 멤버 노드는 동일한 파일 집합에 액세스합니다.

동시성

캐시 일관성은 데이터 일관성과 무결성을 보장하기 위해 클러스터형 파일 시스템의 핵심입니다. 클러스터에 있는 모든 파일의 단일 버전이 클러스터 내의 모든 노드에 표시되어야 합니다. 파일 시스템은 클러스터의 멤버가 동일한 스토리지 블록을 동시에 업데이트하고 데이터 손상을 유발하지 못하도록 해야 합니다. 이를 위해 공유 스토리지 파일 시스템은 클러스터 전체 잠금 메커니즘을 사용하여 스토리지에 대한 액세스를 동시성 제어 메커니즘으로 사용합니다. 예를 들어 새 파일을 만들거나 여러 서버에서 열려 있는 파일에 쓰기 전에 서버의 파일 시스템 구성 요소가 올바른 잠금을 가져와야 합니다.

클러스터 파일 시스템의 요구 사항은 **Apache** 웹 서버와 같은 고가용성 서비스를 제공하는 것입니다. 클러스터의 모든 멤버는 공유 디스크 파일 시스템에 저장된 데이터의 전체 일관성을 확인하고 모든 업데이트는 잠금 메커니즘에 의해 올바르게 중재됩니다.

성능 특성

공유 디스크 파일 시스템은 잠금 오버헤드의 계산 비용으로 인해 동일한 시스템에서 실행되는 로컬 파일 시스템뿐만 아니라 항상 작동하지 않습니다. 공유 디스크 파일 시스템은 각 노드가 다른 노드와 공유되지 않는 특정 파일 집합 또는 노드 집합에서 거의 독점적으로 읽기 전용 방식으로 공유되는 특정 파일 집합에 거의 독점적으로 쓰는 워크로드에서 잘 수행됩니다. 이로 인해 최소 노드 간 캐시가 무효화되고 성능을 극대화할 수 있습니다.

공유 디스크 파일 시스템을 설정하는 것은 복잡하며 공유 디스크 파일 시스템에서 제대로 작동하도록 애플리케이션을 튜닝하는 것은 어려울 수 있습니다.

사용 가능한 공유 스토리지 파일 시스템

- Red Hat Enterprise Linux는 GFS2 파일 시스템을 제공합니다. GFS2는 Red Hat Enterprise Linux 고가용성 애드온 및 복구 스토리지 애드온과 긴밀하게 통합됩니다.

Red Hat Enterprise Linux는 2개에서 16개 노드까지의 규모에 해당하는 클러스터에서 GFS2를 지원합니다.

1.9. 네트워크 및 공유 스토리지 파일 시스템 중에서 선택

네트워크 및 공유 스토리지 파일 시스템 중에서 선택할 때 다음 사항을 고려하십시오.

- NFS 기반 네트워크 파일 시스템은 NFS 서버를 제공하는 환경에서 매우 일반적이고 인기 있는 선택 사항입니다.
- 네트워크 파일 시스템은 Infiniband 또는 10 기가비트 이더넷과 같은 고성능 네트워킹 기술을 사용하여 배포할 수 있습니다. 즉, 스토리지에 원시 대역폭을 얻으려면 공유 스토리지 파일 시스템으로 전환해서는 안 됩니다. 액세스 속도가 매우 중요한 경우 NFS를 사용하여 XFS와 같은 로컬 파일 시스템을 내보냅니다.
- 공유 스토리지 파일 시스템은 설정 또는 유지 관리가 쉽지 않으므로 필요한 가용성을 로컬 또는 네트워크 파일 시스템으로 제공할 수 없는 경우에만 배포해야 합니다.
- 클러스터된 환경의 공유 스토리지 파일 시스템을 사용하면 고가용성 서비스 재배치와 관련된 일반적인 장애 해결 시나리오 중에 수행해야 하는 단계를 제거하여 다운타임을 줄일 수 있습니다.

공유 스토리지 파일 시스템에 대한 구체적인 사용 사례가 없는 한 네트워크 파일 시스템을 사용하는 것이 좋습니다. 공유 스토리지 파일 시스템은 주로 다운타임을 최소화하고 엄격한 서비스 수준 요구 사항을 갖춘 고가용성 서비스를 제공해야 하는 배포에 사용됩니다.

1.10. 볼륨 관리 파일 시스템

볼륨 관리 파일 시스템은 단순성 및 스택 내 최적화를 위해 전체 스토리지 스택을 통합합니다.

사용 가능한 볼륨 관리 파일 시스템

-

Red Hat Enterprise Linux 9는 **Stratis** 볼륨 관리자를 기술 프리뷰로 제공합니다. **Stratis**는 **XFS**를 파일 시스템 계층에 사용하고 **LVM,** 장치 매퍼 및 기타 구성 요소와 통합합니다.

Stratis는 **Red Hat Enterprise Linux 8.0**에서 처음 릴리스되었습니다. **Red Hat**은 **Red Hat**이 더 이상 사용되지 않을 때 생성된 간격을 채우기 위해 사용됩니다. **Stratis 1.0**은 사용자에게 복잡성을 숨기는 동안 상당한 스토리지 관리 작업을 수행할 수 있는 직관적인 명령줄 기반 볼륨 관리자입니다.

- 볼륨 관리
- 풀 생성
- 씬 스토리지 풀
- 스냅샷
- 자동화된 읽기 캐시

Stratis는 강력한 기능을 제공하지만 **currently lacks certain capabilities of other offerings that it might be compared to, such as RuntimeClass** 또는 **RuntimeClass.** 대부분의 경우 자체 복구로 **CRC**를 지원하지 않습니다.

2장. RHEL 시스템 역할을 사용하여 로컬 스토리지 관리

Ansible을 사용하여 LVM 및 로컬 파일 시스템(FS)을 관리하려면 RHEL 9에서 사용할 수 있는 RHEL 시스템 역할 중 하나인 스토리지 역할을 사용할 수 있습니다.

스토리지 역할을 사용하면 RHEL 7.7부터 여러 시스템의 디스크 및 논리 볼륨 및 모든 버전의 RHEL에서 파일 시스템을 자동으로 관리할 수 있습니다.

RHEL 시스템 역할 및 해당 역할을 적용하는 방법에 대한 자세한 내용은 RHEL 시스템 역할 소개를 참조하십시오.

2.1. 스토리지 RHEL 시스템 역할 소개

스토리지 역할은 다음을 관리할 수 있습니다.

- 분할되지 않은 디스크의 파일 시스템
- 논리 볼륨 및 파일 시스템을 포함한 LVM 볼륨 그룹 완료
- MD RAID 볼륨 및 파일 시스템

스토리지 역할을 사용하면 다음 작업을 수행할 수 있습니다.

- 파일 시스템 생성
- 파일 시스템 제거
- 파일 시스템 마운트
- 파일 시스템 마운트 해제

- LVM 볼륨 그룹 만들기
- LVM 볼륨 그룹 제거
- 논리 볼륨 생성
- 논리 볼륨 제거
- RAID 볼륨 생성
- RAID 볼륨 제거
- RAID를 사용하여 LVM 볼륨 그룹 생성
- RAID를 사용하여 LVM 볼륨 그룹 제거
- 암호화된 LVM 볼륨 그룹 만들기
- RAID를 사용하여 LVM 논리 볼륨 생성

2.2. 스토리지 RHEL 시스템 역할에서 스토리지 장치를 식별하는 매개변수

스토리지 역할 구성은 다음 변수에 나열된 파일 시스템, 볼륨 및 풀에만 영향을 미칩니다.

storage_volumes

관리할 모든 파티션되지 않은 디스크의 파일 시스템 목록입니다.

storage_volumes 에는 잘못된 볼륨도 포함 할 수 있습니다.

파티션은 현재 지원되지 않습니다.

storage_pools

관리할 풀 목록입니다.

현재 지원되는 유일한 풀 유형은 **LVM**입니다. **LVM**을 사용하면 풀이 볼륨 그룹(**VG**)을 나타냅니다. 각 풀에는 역할에서 관리할 볼륨 목록이 있습니다. **LVM**을 사용하면 각 볼륨이 파일 시스템의 논리 볼륨(**LV**)에 해당합니다.

2.3. 예제 ANSIBLE PLAYBOOK 블록 장치에 XFS 파일 시스템을 생성

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 기본 매개 변수를 사용하여 블록 장치에 **XFS** 파일 시스템을 생성하도록 **storage** 역할을 적용합니다.

주의

스토리지 역할은 분할되지 않은 전체 디스크 또는 논리 볼륨(**LV**)에서만 파일 시스템을 생성할 수 있습니다. 파티션에 파일 시스템을 만들 수 없습니다.

예 2.1. /dev/sdb에서 XFS를 생성하는 플레이북

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
  roles:
    - rhel-system-roles.storage
```

- 볼륨 이름(예의 ***barefs***)은 현재 임의로 사용할 수 있습니다. 스토리지 역할은 **disks:** 속성 아래에 나열된 디스크 장치에서 볼륨을 식별합니다.

- XFS는 RHEL 9의 기본 파일 시스템이므로 fs_type: xfs 행을 생략할 수 있습니다.

- LV에 파일 시스템을 생성하려면 enclosing 볼륨 그룹을 포함하여 disks: 속성 아래에 LVM 설정을 제공합니다. 자세한 내용은 Example Ansible Playbook to manage logical volumes 에서 참조하십시오.

  LV 장치의 경로를 제공하지 마십시오.

추가 리소스

- /usr/share/ansible/roles/rhel-system-roles.storage/README.md 파일.

2.4. 파일 시스템을 영구적으로 마운트하는 ANSIBLE 플레이북의 예

이 섹션에서는 예제 Ansible 플레이북을 제공합니다. 이 플레이북은 storage 역할을 적용하여 XFS 파일 시스템을 즉시 지속적으로 마운트합니다.

예 2.2. /dev/sdb에 파일 시스템을 /mnt/data에 마운트하는 플레이북

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- 이 Playbook은 /etc/fstab 파일에 파일 시스템을 추가하고 파일 시스템을 즉시 마운트합니다.

- /dev/sdb 장치 또는 마운트 지점 디렉터리의 파일 시스템이 존재하지 않는 경우 플레이북에서 해당 시스템을 생성합니다.

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일.

2.5. 논리 볼륨을 관리하는 ANSIBLE 플레이북 예

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 스토리지 역할을 적용하여 볼륨 그룹에 **LVM** 논리 볼륨을 생성합니다.

예 2.3. myvg 볼륨 그룹에 mylv 논리 볼륨을 생성하는 플레이북

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

- **myvg** 볼륨 그룹은 다음 디스크로 구성됩니다.
  - **/dev/sda**
  - **/dev/sdb**
  - **/dev/sdc**
- **myvg** 볼륨 그룹이 이미 있는 경우 **Playbook**은 볼륨 그룹에 논리 볼륨을 추가합니다.
- **myvg** 볼륨 그룹이 없으면 플레이북에서 해당 그룹을 생성합니다.
- 이 플레이북은 **mylv** 논리 볼륨에 **Ext4** 파일 시스템을 생성하고 **/mnt** 에 파일 시스템을 영

구적으로 마운트합니다.

추가 리소스

- /usr/share/ansible/roles/rhel-system-roles.storage/README.md 파일.

2.6. 온라인 블록 삭제 활성화를 위한 ANSIBLE PLAYBOOK 예

이 섹션에서는 예제 Ansible 플레이북을 제공합니다. 이 Playbook은 온라인 블록 삭제가 활성화된 XFS 파일 시스템을 마운트하는 스토리지 역할을 적용합니다.

예 2.4. /mnt/data/에서 온라인 블록 삭제를 활성화하는 플레이북

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
        mount_options: discard
  roles:
    - rhel-system-roles.storage
```

추가 리소스

- Ansible 플레이북의 예제로 파일 시스템을 지속적으로 마운트합니다.
- /usr/share/ansible/roles/rhel-system-roles.storage/README.md 파일.

2.7. 예시 ANSIBLE PLAYBOOK: EXT4 파일 시스템을 생성 및 마운트

이 섹션에서는 예제 Ansible 플레이북을 제공합니다. 이 플레이북은 storage 역할을 적용하여 Ext4 파일 시스템을 생성하고 마운트합니다.

예 2.5. /dev/sdb에 Ext4를 생성하고 /mnt/data에 마운트하는 플레이북

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: ext4
        fs_label: label-name
        mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- **Playbook**은 **/dev/sdb** 디스크에 파일 시스템을 생성합니다.
- 플레이북은 ***/mnt/data*** 디렉터리에 파일 시스템을 영구적으로 마운트합니다.
- 파일 시스템의 레이블은 ***label-name*** 입니다.

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일.

2.8. EXT3 파일 시스템을 생성하고 마운트하는 예제 ANSIBLE PLAYBOOK

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 **storage** 역할을 적용하여 **Ext3** 파일 시스템을 생성하고 마운트합니다.

예 2.6. /dev/sdb 에 Ext3를 생성하여 /mnt/data에 마운트하는 플레이북

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: ext3
        fs_label: label-name
```

```
    mount_point: /mnt/data
roles:
  - rhel-system-roles.storage
```

- **Playbook은 /dev/sdb 디스크에 파일 시스템을 생성합니다.**
- **플레이북은 */mnt/data* 디렉터리에 파일 시스템을 영구적으로 마운트합니다.**
- **파일 시스템의 레이블은 *label-name* 입니다.**

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md 파일.**

2.9. 스토리지 RHEL 시스템 역할을 사용하여 기존 EXT4 또는 EXT3 파일 시스템의 크기를 조정하는 ANSIBLE PLAYBOOK의 예

이 섹션에서는 예제 Ansible 플레이북을 제공합니다. 이 Playbook은 스토리지 역할을 적용하여 블록 장치의 기존 Ext4 또는 Ext3 파일 시스템의 크기를 조정합니다.

예 2.7. 디스크에 단일 볼륨을 설정하는 플레이북

```
---
- name: Create a disk device mounted on /opt/barefs
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - /dev/sdb
        size: 12 GiB
        fs_type: ext4
        mount_point: /opt/barefs
  roles:
    - rhel-system-roles.storage
```

- **이전 예제의 볼륨이 이미 있는 경우 볼륨 크기를 조정하려면 매개 변수 크기에 다른 값을 사용하여 동일한 플레이북을 실행해야 합니다. 예를 들면 다음과 같습니다.**

-

예 2.8. /dev/sdb에서 ext4 의 크기를 조정하는 플레이북

```
---
- name: Create a disk device mounted on /opt/barefs
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - /dev/sdb
        size: 10 GiB
        fs_type: ext4
        mount_point: /opt/barefs
  roles:
    - rhel-system-roles.storage
```

- 볼륨 이름(예의**barefs**)은 현재 임의로입니다. **Storage** 역할은 **disks:** 속성 아래에 나열된 디스크 장치에서 볼륨을 식별합니다.

참고

다른 파일 시스템의 작업 크기 조정을 사용하면 작업 중인 장치의 데이터가 손상될 수 있습니다.

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일.

2.10. 스토리지 RHEL 시스템 역할을 사용하여 LVM에서 기존 파일 시스템의 크기를 조정하는 ANSIBLE PLAYBOOK의 예

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 스토리지 **RHEL** 시스템 역할을 적용하여 파일 시스템으로 **LVM** 논리 볼륨의 크기를 조정합니다.

주의

다른 파일 시스템의 작업 크기 조정을 사용하면 작업 중인 장치의 데이터가 손상될 수 있습니다.

예 2.9. myvg 볼륨 그룹의 기존 mylv1 및 myvl2 논리 볼륨의 크기를 조정하는 플레이북

```
---

- hosts: all
  vars:
    storage_pools:
      - name: myvg
        disks:
          - /dev/sda
          - /dev/sdb
          - /dev/sdc
        volumes:
          - name: mylv1
            size: 10 GiB
            fs_type: ext4
            mount_point: /opt/mount1
          - name: mylv2
            size: 50 GiB
            fs_type: ext4
            mount_point: /opt/mount2

- name: Create LVM pool over three disks
  include_role:
    name: rhel-system-roles.storage
```

- 이 **Playbook**은 다음과 같은 기존 파일 시스템의 크기를 조정합니다.
  - **/opt/mount1** 에 마운트된 **mylv1** 볼륨의 **Ext4** 파일 시스템은 **10GiB**로 크기를 조정합니다.
  - **/opt/mount2** 에 마운트된 **mylv2** 볼륨의 **Ext4** 파일 시스템은 **50GiB**로 크기를 조정합니다.

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일.

2.11. 스토리지 RHEL 시스템 역할을 사용하여 스왑 볼륨을 생성하는 ANSIBLE PLAYBOOK의 예

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 스토리지 역할을 적용하여 스왑 볼륨이 없는 경우 또는 기본 매개변수를 사용하는 블록 장치에 스왑 볼륨을 수정하거나 스왑 볼륨이 이미 존재하는 경우 수정합니다.

예 2.10. /dev/sdb에서 기존 XFS를 생성하거나 수정하는 플레이북

```
---
- name: Create a disk device with swap
- hosts: all
  vars:
    storage_volumes:
      - name: swap_fs
        type: disk
        disks:
          - /dev/sdb
  size: 15 GiB
        fs_type: swap
  roles:
    - rhel-system-roles.storage
```

- 볼륨 이름(예의 ***swap_fs***)은 현재 임의적입니다. 스토리지 역할은 **disks:** 속성 아래에 나열된 디스크 장치에서 볼륨을 식별합니다.

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일.

2.12. 스토리지 시스템 역할을 사용하여 RAID 볼륨 구성

스토리지 시스템 역할을 사용하면 **Red Hat Ansible Automation Platform** 및 **Ansible-Core**를 사용하여 **RHEL**에서 **RAID** 볼륨을 구성할 수 있습니다. 매개변수를 사용하여 **Ansible** 플레이북을 생성하여 요구 사항에 맞게 **RAID** 볼륨을 구성합니다.

사전 요구 사항

- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.

- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.

- 스토리지 시스템 역할을 사용하여 RAID 볼륨을 배포하려는 시스템을 자세히 설명하는 인벤토리 파일이 있습니다.

절차

1. 다음 콘텐츠를 사용하여 새 *playbook.yml* 파일을 생성합니다.

```
---
- name: Configure the storage
  hosts: managed-node-01.example.com
  tasks:
  - name: Create a RAID on sdd, sde, sdf, and sdg
    include_role:
      name: rhel-system-roles.storage
    vars:
    storage_safe_mode: false
    storage_volumes:
      - name: data
        type: raid
        disks: [sdd, sde, sdf, sdg]
        raid_level: raid0
        raid_chunk_size: 32 KiB
        mount_point: /mnt/data
        state: present
```

주의

장치 이름은 예를 들어 시스템에 새 디스크를 추가하는 경우와 같이 특정 상황에서 변경될 수 있습니다. 따라서 데이터 손실을 방지하려면 플레이북에서 특정 디스크 이름을 사용하지 마십시오.

2. 선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3. 플레이북을 실행합니다.

```
# ansible-playbook -i inventory.file /path/to/file/playbook.yml
```

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일
- **RHEL System Roles**를 사용하도록 제어 노드 및 관리형 노드 준비

2.13. 스토리지 RHEL 시스템 역할을 사용하여 RAID로 LVM 풀 구성

스토리지 시스템 역할을 사용하면 **Red Hat Ansible Automation Platform**을 사용하여 **RHEL**에서 **RAID**로 **LVM** 풀을 구성할 수 있습니다. 이 섹션에서는 사용 가능한 매개 변수를 사용하여 **Ansible** 플레이북을 설정하여 **RAID**로 **LVM** 풀을 구성하는 방법을 배웁니다.

사전 요구 사항

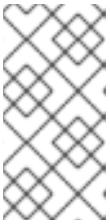
- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.
- 스토리지 시스템 역할을 사용하여 **RAID**로 **LVM** 풀을 구성하려는 시스템을 자세히 설명하는 인벤토리 파일이 있습니다.

절차

1. 다음 내용으로 새 ***playbook.yml*** 파일을 생성합니다.

```
- hosts: all
  vars:
    storage_safe_mode: false
    storage_pools:
      - name: my_pool
        type: lvm
        disks: [sdh, sdi]
        raid_level: raid1
        volumes:
          - name: my_pool
            size: "1 GiB"
            mount_point: "/mnt/app/shared"
            fs_type: xfs
```

```
        state: present
  roles:
    - name: rhel-system-roles.storage
```



참고

RAID를 사용하여 **LVM** 풀을 생성하려면 **raid_level** 매개변수를 사용하여 **RAID** 유형을 지정해야 합니다.

2. 선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3. 인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory.file /path/to/file/playbook.yml
```

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일.

2.14. 스토리지 RHEL 시스템 역할을 사용하여 LVM에서 VDO 볼륨을 압축하고 중복 제거하는 ANSIBLE 플레이북의 예

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 스토리지 **RHEL** 시스템 역할을 적용하여 **VDO(**가상 데이터 최적화 도구**)**를 사용하여 **LVM(Logical Volumes)** 압축 및 중복 제거를 활성화합니다.

예 2.11. my vg 볼륨 그룹에서 **mylv1 LVM VDO** 볼륨을 생성하는 플레이북

```
---
- name: Create LVM VDO volume under volume group 'myvg'
  hosts: all
  roles:
    -rhel-system-roles.storage
  vars:
    storage_pools:
     - name: myvg
       disks:
         - /dev/sdb
       volumes:
         - name: mylv1
```

```
    compression: true
    deduplication: true
    vdo_pool_size: 10 GiB
    size: 30 GiB
    mount_point: /mnt/app/shared
```

이 예제에서는 압축 및 중복 제거 풀이 **true**로 설정되어 **VDO**가 사용되도록 지정합니다. 다음은 이러한 매개변수의 사용을 설명합니다.

- 중복 제거 는 스토리지 볼륨에 저장된 중복된 데이터를 중복 제거하는 데 사용됩니다.
- 압축은 스토리지 볼륨에 저장된 데이터를 압축하는 데 사용되어 스토리지 용량이 증가합니다.
- **vdo_pool_size**는 장치에서 사용하는 실제 크기를 지정합니다. **VDO** 볼륨의 가상 크기는 **size** 매개 변수에 의해 설정됩니다. 알림: **LVM VDO**의 스토리지 역할 때문에 풀당 하나의 볼륨만 압축 및 중복 제거를 사용할 수 있습니다.

2.15. 스토리지 RHEL 시스템 역할을 사용하여 LUKS 암호화 볼륨 생성

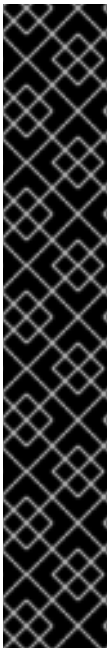
storage 역할을 사용하여 **Ansible** 플레이북을 실행하여 **LUKS**로 암호화된 볼륨을 생성하고 구성할 수 있습니다.

사전 요구 사항

- **crypto_policies** 시스템 역할로 설정하려는 하나 이상의 *관리형 노드에* 대한 액세스 및 권한.
- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 제어 노드에 대한 액세스 및 권한.

  제어 노드에서 다음을 수행합니다.

  - **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.



중요

RHEL 8.0-8.5는 **Ansible** 기반 자동화를 위해 **Ansible Engine 2.9**가 포함된 별도의 **Ansible** 리포지토리에 대한 액세스를 제공했습니다. **Ansible Engine**에는 **ansible , ansible -playbook, docker** 및 **podman** 과 같은 커넥터, 여러 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. **Ansible Engine**을 확보하고 설치하는 방법에 대한 자세한 내용은 **Red Hat Ansible Engine** 지식베이스를 다운로드하고 설치하는 방법 문서를 참조하십시오.

RHEL 8.6 및 **9.0**에서는 **Ansible** 명령줄 유틸리티, 명령 및 소규모의 기본 제공 **Ansible** 플러그인 세트가 포함된 **Ansible Core(ansible-core** 패키지로 제공)를 도입했습니다. **RHEL**은 **AppStream** 리포지토리를 통해 이 패키지를 제공하며 제한된 지원 범위를 제공합니다. 자세한 내용은 **RHEL 9** 및 **RHEL 8.6** 이상 **AppStream** 리포지토리 지식 베이스에 포함된 **Ansible Core** 패키지에 대한 지원 범위를 참조하십시오.

- 관리 노드를 나열하는 인벤토리 파일.

절차

1. 다음 내용으로 새 ***playbook.yml*** 파일을 생성합니다.

```
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        fs_label: label-name
        mount_point: /mnt/data
        encryption: true
        encryption_password: your-password
  roles:
    - rhel-system-roles.storage
```

2. 선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3. 인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory.file /path/to/file/playbook.yml
```

추가 리소스

- /usr/share/ansible/roles/rhel-system-roles.storage/README.md file

2.16. 스토리지 RHEL 시스템 역할을 사용하여 백분율로 풀 볼륨 크기를 표시하는 ANSIBLE 플레이북의 예

이 섹션에서는 예제 Ansible 플레이북을 제공합니다. 이 플레이북은 스토리지 시스템 역할을 적용하여 풀의 총 크기의 백분율로 논리 관리자 볼륨(LVM) 볼륨 크기를 표시할 수 있습니다.

예 2.12. 볼륨 크기를 풀 총 크기의 백분율로 표시하는 플레이북

```
---
- name: Express volume sizes as a percentage of the pool's total size
  hosts: all
  roles
    - rhel-system-roles.storage
  vars:
    storage_pools:
    - name: myvg
      disks:
        - /dev/sdb
      volumes:
        - name: data
          size: 60%
          mount_point: /opt/mount/data
        - name: web
          size: 30%
          mount_point: /opt/mount/web
        - name: cache
          size: 10%
          mount_point: /opt/cache/mount
```

이 예에서는 LVM 볼륨의 크기를 풀 크기의 백분율로 지정합니다. 예를 들면 다음과 같습니다. "60%". 또한 LVM 볼륨의 크기를 사람이 읽을 수 있는 파일 시스템 크기(예: "10g" 또는 "50GiB)"의 풀 크기의 백분율로 지정할 수도 있습니다.

2.17. 추가 리소스

- /usr/share/doc/rhel-system-roles/storage/

- **/usr/share/ansible/roles/rhel-system-roles.storage/**

3장. NFS 공유 마운트

시스템 관리자는 시스템에 원격 NFS 공유를 마운트하여 공유 데이터에 액세스할 수 있습니다.

3.1. NFS 소개

이 섹션에서는 NFS 서비스의 기본 개념에 대해 설명합니다.

NFS(네트워크 파일 시스템)를 사용하면 원격 호스트에서 네트워크를 통해 파일 시스템을 마운트하고 로컬에 마운트된 것처럼 해당 파일 시스템과 상호 작용할 수 있습니다. 이를 통해 네트워크의 중앙 집중식 서버에 리소스를 통합할 수 있습니다.

NFS 서버는 /etc/exports 구성 파일을 참조하여 클라이언트가 내보낸 모든 파일 시스템에 액세스할 수 있는지 여부를 확인합니다. 확인되면 모든 파일 및 디렉터리 작업을 사용자가 사용할 수 있습니다.

3.2. 지원되는 NFS 버전

이 섹션에는 Red Hat Enterprise Linux에서 지원되는 NFS 버전과 해당 기능이 나열되어 있습니다.

현재 Red Hat Enterprise Linux 9는 다음과 같은 주요 버전의 NFS를 지원합니다.

- NFS 버전 3(NFSv3)은 안전한 비동기 쓰기를 지원하며 이전 NFSv2보다 오류 처리 시 더 강력합니다. 또한 64비트 파일 크기 및 오프셋을 지원하여 클라이언트가 2GB 이상의 파일 데이터에 액세스할 수 있습니다.
- NFS 버전 4(NFSv4)는 방화벽 및 인터넷에서 작동하므로 더 이상 EgressIP bind 서비스가 필요하지 않고 ACL(액세스 제어 목록)을 지원하고 상태 저장 작업을 활용합니다.

NFS 버전 2(NFSv2)는 Red Hat에서 더 이상 지원하지 않습니다.

기본 NFS 버전

Red Hat Enterprise Linux 9의 기본 NFS 버전은 4.2입니다. NFS 클라이언트는 기본적으로 NFSv4.2 사용을 시도하고 서버가 NFSv4.2를 지원하지 않는 경우 NFSv4.1로 돌아갑니다. 나중에 마운트가 NFSv4.0으로 대체된 다음 NFSv3로 돌아갑니다.

마이너 NFS 버전의 기능

다음은 Red Hat Enterprise Linux 9의 NFSv4.2의 기능입니다.

서버 측 복사

NFS 클라이언트가 copy_file_range() 시스템 호출을 사용하여 네트워크 리소스를 낭비하지 않고 데이터를 효율적으로 복사할 수 있습니다.

스파스 파일

파일에 0es만 구성된 할당되지 않았거나 초기화되지 않은 데이터 블록이 하나 이상 있을 수 있습니다. *NFSv4.2의 lseek() 작업은 search _hole() 및 seek_data() 를 지원하므로 애플리케이션이 스파스 파일에 있는 홀의 위치를 매핑할 수 있습니다.*

공간 예약

스토리지 서버가 여유 공간을 예약할 수 있도록 허용하여 서버가 공간이 부족해지는 것을 방지합니다. NFSv4.2에서는 allocate() 작업을 지원하여 공간을 예약하고, 공간을 할당 해제하는 deallocate() 작업, 파일에서 공간을 미리 할당하거나 할당 해제하는 fallocate() 작업을 지원합니다.

레이블이 지정된 NFS

데이터 액세스 권한을 적용하고 NFS 파일 시스템의 개별 파일에 대해 클라이언트와 서버 간에 SELinux 레이블을 활성화합니다.

레이아웃 개선 사항

일부 Parallel NFS(pNFS) 서버가 더 나은 성능 통계를 수집할 수 있도록 하는 layoutstats() 작업을 제공합니다.

다음은 NFSv4.1의 기능입니다.

- *네트워크의 성능과 보안을 강화하며 pNFS에 대한 클라이언트 측 지원도 포함합니다.*
- *더 이상 콜백에 대한 별도의 TCP 연결이 필요하지 않으므로, 예를 들어 NAT 또는 방화벽이 방해하는 경우도 NFS 서버에서 클라이언트에 연결할 수 없는 경우에도 NFS 서버에서 위임을 부여할 수 있습니다.*
- *재부팅 작업을 제외하고 정확히 한 번만 의미 체계를 제공하므로, 일부 작업에서 응답이 손실되고 작업을 두 번 전송한 경우 특정 작업에서 부정확한 결과를 반환하는 경우가 있습니다.*

3.3. NFS에 필요한 서비스

이 섹션에는 NFS 서버를 실행하거나 NFS 공유를 마운트하는 데 필요한 시스템 서비스가 나열되어 있습니다. Red Hat Enterprise Linux는 이러한 서비스를 자동으로 시작합니다.

Red Hat Enterprise Linux는 커널 수준 지원 및 서비스 프로세스를 결합하여 NFS 파일 공유를 제공합니다. 모든 NFS 버전은 클라이언트와 서버 간의 RPC(Remote Procedure calls)에 의존합니다. NFS 파일 시스템을 공유하거나 마운트하기 위해 다음 서비스가 구현되는 NFS 버전에 따라 함께 작동합니다.

nfsd

공유 NFS 파일 시스템에 대해 요청하는 NFS 서버 커널 모듈.

rpcbind

로컬 RPC 서비스의 포트 예약을 허용합니다. 그런 다음 이러한 포트를 사용할 수 있도록 (또는 알림) 해당 원격 RPC 서비스에 액세스할 수 있습니다. E gressIPbind 서비스는 RPC 서비스에 대한 요청에 응답하고 요청된 RPC 서비스에 대한 연결을 설정합니다. NFSv4에서는 사용되지 않습니다.

rpc.mountd

이 프로세스는 NFS 서버에서 NFSv3 클라이언트의 MOUNT 요청을 처리하는 데 사용됩니다. 요청된 NFS 공유를 현재 NFS 서버에서 내보내고 클라이언트가 액세스할 수 있는지 확인합니다. 마운트 요청이 허용되는 경우 nfs-mountd 서비스는 Success 상태로 응답하고 이 NFS 공유에 대한 File-Handle을 다시 NFS 클라이언트로 제공합니다.

rpc.nfsd

이 프로세스를 통해 서버가 광고하는 명시적인 NFS 버전 및 프로토콜을 사용할 수 있습니다. NFS 클라이언트가 연결할 때마다 서버 스레드 제공 등 NFS 클라이언트의 동적 요구를 충족하기 위해 Linux 커널과 함께 작동합니다. 이 프로세스는 nfs-server 서비스에 해당합니다.

lockd

이는 클라이언트와 서버에서 모두 실행되는 커널 스레드입니다. NFSv3 클라이언트가 서버의 파일을 잠글 수 있도록 하는 NLM(Network Lock Manager) 프로토콜을 구현합니다. NFS 서버를 실행하고 NFS 파일 시스템이 마운트될 때마다 자동으로 시작됩니다.

rpc.statd

이 프로세스는 NFV(Network Status Monitor) RPC 프로토콜을 구현하여 NFS 서버가 정상적으로 종료되지 않고 다시 시작될 때 NFS 클라이언트에 알립니다. RuntimeClass-statd 서비스는 nfs-server 서비스에 의해 자동으로 시작되며 사용자 구성이 필요하지 않습니다. NFSv4에서는 사용되지 않습니다.

rpc.rquotad

이 프로세스는 원격 사용자에게 사용자 할당량 정보를 제공합니다. quota-rpc 패키지에서 제공하는 RuntimeClass-rquotad 서비스는 nfs-server 가 시작될 때 사용자가 시작해야 합니다.

rpc.idmapd

이 프로세스는 NFSv4 클라이언트 및 서버 백업 호출을 제공합니다. 이 호출은 사용자@도메인의 형식의 문자열과 로컬 UID 및 GID 사이에 매핑됩니다. idmapd 가 NFSv4에서 작동하려면 /etc/idmapd.conf 파일을 구성해야 합니다. 최소한 NFSv4 매핑 도메인을 정의하는 Domain 매개 변수를 지정해야 합니다. NFSv4 매핑 도메인이 DNS 도메인 이름과 동일한 경우 이 매개 변수를 건너뛸 수 있습니다. ID 매핑이 제대로 작동하려면 클라이언트 및 서버는 NFSv4 매핑 도메인에 동의해야 합니다.

NFSv4 서버만 nfs- idmapd 서비스에 의해 시작되는 RuntimeClass. idmapd를 사용합니다. NFSv4 클라이언트는 ID 매핑을 수행하기 위해 커널 온 디맨드에서 호출하는 인증 키 기반 nfsidmap 유틸리티를 사용합니다. nfsidmap 에 문제가 있는 경우 클라이언트는 Jenkinsfile .idmapd를 다시 사용합니다.

NFSv4를 사용한 RPC 서비스

NFSv4 프로토콜의 마운트 및 잠금 프로토콜이 통합되었습니다. 서버는 또한 잘 알려진 TCP 포트 2049에서 수신 대기합니다. 따라서 NFSv4는 EgressIP bind,lockd, RuntimeClass -statd 서비스와 상호 작용할 필요가 없습니다. nfs-mountd 서비스는 내보내기를 설정하기 위해 NFS 서버에 계속 필요하지만 over-the-wire 작업에는 포함되지 않습니다.

추가 리소스

- Diffie bind를 사용하지 않고 NFSv4만 구성.

3.4. NFS 호스트 이름 형식

이 섹션에서는 NFS 공유를 마운트하거나 내보낼 때 호스트를 지정하는 데 사용할 수 있는 다양한 형식을 설명합니다.

다음 형식으로 호스트를 지정할 수 있습니다.

단일 머신

다음 중 하나를 수행합니다.

- 정규화된 도메인 이름(서버에서 확인할 수 있음)

- *호스트 이름(서버에서 확인할 수 있음)*

- ***IP** 주소.*

***IP** 네트워크*

다음 형식 중 하나가 유효합니다.

- ***a.b.c.d /z.b.c.d**는 네트워크이고 **z** 는 넷마스크의 비트 수입니다(예:**DNS**).*

- ***a.b.c.d /netmask.b.c.d**는 네트워크 및 넷마스크 입니다(예: **192.168.100.8/255.255.255.0**).*

netgroups

@group-name 형식 . 여기서 group-name 은 NIS netgroup 이름입니다.

3.5. NFS 설치

*이 절차에서는 **NFS** 공유를 마운트하거나 내보내는 데 필요한 모든 패키지를 설치합니다.*

절차

- ***nfs-utils** 패키지를 설치합니다.*

```
# dnf install nfs-utils
```

3.6. NFS 내보내기 검색

*이 절차에서는 지정된 **NFSv3** 또는 **NFSv4** 서버 내보내기에 사용할 파일 시스템을 검색합니다.*

절차

- ***NFSv3**를 지원하는 모든 서버에서 **showmount** 유틸리티를 사용합니다.*

-

```
$ showmount --exports my-server

Export list for my-server
/exports/foo
/exports/bar
```

- **NFSv4**를 지원하는 서버를 사용하여 루트 디렉토리를 마운트하고 다음을 확인합니다.

```
# mount my-server:/ /mnt/
# ls /mnt/

exports

# ls /mnt/exports/

foo
bar
```

NFSv4 및 **NFSv3**를 모두 지원하는 서버에서 두 방법 모두 작동하고 동일한 결과를 제공합니다.

추가 리소스

- **showmount(8)** 도움말 페이지

3.7. MOUNT를 사용하여 NFS 공유 마운트

이 절차에서는 **mount** 유틸리티를 사용하여 서버에서 내보낸 **NFS** 공유를 마운트합니다.

절차

- **NFS** 공유를 마운트하려면 다음 명령을 사용하십시오.

```
# mount -t nfs -o options host:/remote/export /local/directory
```

이 명령은 다음 변수를 사용합니다.

옵션

쉼표로 구분된 마운트 옵션 목록입니다.

호스트

마운트하려는 파일 시스템을 내보내는 서버의 호스트 이름, IP 주소 또는 정규화된 도메인 이름입니다.

/remote/export

서버에서 내보낼 파일 시스템 또는 디렉터리, 즉 마운트하려는 디렉터리입니다.

/local/directory

/remote/export 가 마운트된 클라이언트 위치입니다.

추가 리소스

- *일반적인 NFS 마운트 옵션.*
- *NFS 호스트 이름 형식.*
- *마운트를 사용하여 파일 시스템 마운트.*
- *mount(8) 도움말 페이지*
- *export(5) 도움말 페이지*

3.8. 일반적인 NFS 마운트 옵션

이 섹션에는 NFS 공유를 마운트할 때 일반적으로 사용되는 옵션이 나열되어 있습니다. 이러한 옵션은 수동 마운트 명령, /etc/fstab 설정 및 EgressIP 와 함께 사용할 수 있습니다.

lookupcache=mode

커널이 지정된 마운트 지점의 디렉터리 항목 캐시를 관리하는 방법을 지정합니다. 모드에 유효한 인수는 모두,none 또는 positive 입니다.

nfsvers=version

사용할 NFS 프로토콜의 버전을 지정합니다. 여기서 version 은 3,4,4.0,4.1 또는 4.2 입니다. 이는 여러 NFS 서버를 실행하거나 하위 버전으로 마운트 재시도를 비활성화하는 호스트에 유용합니다. 버전을 지정하지 않으면 NFS는 커널 및 마운트 유틸리티에서 지원되는 최고 버전을 사용합니다.

옵션 **vers** 는 **nfsvers** 와 동일하며 호환성 이유로 이 릴리스에 포함되어 있습니다.

noacl

모든 **ACL** 처리를 비활성화합니다. 이는 최신 **ACL** 기술이 이전 시스템과 호환되지 않기 때문에 이전 버전의 **Red Hat Enterprise Linux, Red Hat Linux** 또는 솔라리스와 연결할 때 필요할 수 있습니다.

NOLOCK

파일 잠금을 비활성화합니다. 매우 오래된 **NFS** 서버에 연결할 때 이 설정이 필요합니다.

noexec

마운트된 파일 시스템에서 바이너리의 실행을 방지합니다. 이 명령은 호환되지 않는 바이너리가 포함된 **Linux** 파일 시스템을 마운트할 때 유용합니다.

nosuid

set-user-identifier 및 **set-group-identifier** 비트를 비활성화합니다. 이렇게 하면 원격 사용자가 **setuid** 프로그램을 실행하여 더 높은 권한을 얻지 못하게 됩니다.

port=num

NFS 서버 포트의 숫자 값을 지정합니다. **num** 이 **0** (기본값)이면 사용할 포트 번호에 대해 원격 호스트에 **queries**를 **mount** 합니다. 원격 호스트의 **NFS** 서비스가 **rootfs bind** 서비스에 등록되지 않은 경우 대신 표준 **NFS** 포트 번호 **TCP 2049**가 사용됩니다.

rsize=num 및 **wsize=num**

이러한 옵션은 단일 **NFS** 읽기 또는 쓰기 작업에서 전송할 최대 바이트 수를 설정합니다.

rsize 및 **wsize** 에 대한 고정 기본값이 없습니다. 기본적으로 **NFS**는 서버와 클라이언트 모두 지원하는 가능한 가장 큰 값을 사용합니다. **Red Hat Enterprise Linux 9**에서 클라이언트 및 서버 최대값은 **1,048,576**바이트입니다. 자세한 내용은 **What are the default and maximum values for rsize and wsize with NFS mounts?**를 참조하십시오. **Kbase** 기사입니다.

sec=flavors

마운트된 내보내기의 파일에 액세스하는 데 사용할 보안 플레이버. 플레이버 값은 콜론으로 구분된 하나 이상의 보안 플레이버 목록입니다.

기본적으로 클라이언트는 클라이언트와 서버 모두 지원하는 보안 플레이버를 찾습니다. 서버가 선택한 플레이버를 지원하지 않으면 마운트 작업이 실패합니다.

사용 가능한 플레이버:

- ***head=sys** 는 로컬 **UNIX UID** 및 **GID**를 사용합니다. 이는 **AUTH_SYS** 를 사용하여 **NFS** 작업을 인증합니다.*
- ***et=krb5** 는 로컬 **UNIX UID** 및 **GID** 대신 **Kerberos V5**를 사용하여 사용자를 인증합니다.*
- ***Attack=krb5i** 는 사용자 인증을 위해 **Kerberos V5**를 사용하고 데이터 변조를 방지하기 위해 보안 체크섬을 사용하여 **NFS** 작업을 무결성 검사를 수행합니다.*
- ***europe=krb5p** 는 **Kerberos V5**를 사용하여 사용자 인증, 무결성 검사 및 **NFS** 트래픽을 암호화하여 트래픽 스니핑을 방지합니다. 이는 가장 안전한 설정이지만 성능 오버헤드가 가장 많습니다.*

tcp

***TCP** 프로토콜을 사용하도록 **NFS** 마운트에 지시합니다.*

추가 리소스

- ***mount(8)** 도움말 페이지*
- ***NFS(5)** 도움말 페이지*

3.9. 추가 리소스

- ***Linux NFS why***
- ***NFS** 공유를 영구적으로 마운트합니다.*
- *요청에 따라 **NFS** 공유 마운트.*

4장. NFS 공유 내보내기

시스템 관리자는 NFS 서버를 사용하여 네트워크를 통해 시스템의 디렉터리를 공유할 수 있습니다.

4.1. NFS 소개

이 섹션에서는 NFS 서비스의 기본 개념에 대해 설명합니다.

NFS(네트워크 파일 시스템)를 사용하면 원격 호스트에서 네트워크를 통해 파일 시스템을 마운트하고 로컬에 마운트된 것처럼 해당 파일 시스템과 상호 작용할 수 있습니다. 이를 통해 네트워크의 중앙 집중식 서버에 리소스를 통합할 수 있습니다.

NFS 서버는 /etc/exports 구성 파일을 참조하여 클라이언트가 내보낸 모든 파일 시스템에 액세스할 수 있는지 여부를 확인합니다. 확인되면 모든 파일 및 디렉터리 작업을 사용자가 사용할 수 있습니다.

4.2. 지원되는 NFS 버전

이 섹션에는 Red Hat Enterprise Linux에서 지원되는 NFS 버전과 해당 기능이 나열되어 있습니다.

현재 Red Hat Enterprise Linux 9는 다음과 같은 주요 버전의 NFS를 지원합니다.

- NFS 버전 3(NFSv3)은 안전한 비동기 쓰기를 지원하며 이전 NFSv2보다 오류 처리 시 더 강력합니다. 또한 64비트 파일 크기 및 오프셋을 지원하여 클라이언트가 2GB 이상의 파일 데이터에 액세스할 수 있습니다.
- NFS 버전 4(NFSv4)는 방화벽 및 인터넷에서 작동하므로 더 이상 EgressIP bind 서비스가 필요하지 않고 ACL(액세스 제어 목록)을 지원하고 상태 저장 작업을 활용합니다.

NFS 버전 2(NFSv2)는 Red Hat에서 더 이상 지원하지 않습니다.

기본 NFS 버전

Red Hat Enterprise Linux 9의 기본 NFS 버전은 4.2입니다. NFS 클라이언트는 기본적으로 NFSv4.2 사용을 시도하고 서버가 NFSv4.2를 지원하지 않는 경우 NFSv4.1로 돌아갑니다. 나중에 마운트가 NFSv4.0으로 대체된 다음 NFSv3로 돌아갑니다.

마이너 NFS 버전의 기능

다음은 Red Hat Enterprise Linux 9의 NFSv4.2의 기능입니다.

서버 측 복사

NFS 클라이언트가 copy_file_range() 시스템 호출을 사용하여 네트워크 리소스를 낭비하지 않고 데이터를 효율적으로 복사할 수 있습니다.

스파스 파일

파일에 0es만 구성된 할당되지 않았거나 초기화되지 않은 데이터 블록이 하나 이상 있을 수 있습니다. NFSv4.2의 lseek() 작업은 search _hole() 및 seek_data() 를 지원하므로 애플리케이션이 스파스 파일에 있는 홀의 위치를 매핑할 수 있습니다.

공간 예약

스토리지 서버가 여유 공간을 예약할 수 있도록 허용하여 서버가 공간이 부족해지는 것을 방지합니다. NFSv4.2에서는 allocate() 작업을 지원하여 공간을 예약하고, 공간을 할당 해제하는 deallocate() 작업, 파일에서 공간을 미리 할당하거나 할당 해제하는 fallocate() 작업을 지원합니다.

레이블이 지정된 NFS

데이터 액세스 권한을 적용하고 NFS 파일 시스템의 개별 파일에 대해 클라이언트와 서버 간에 SELinux 레이블을 활성화합니다.

레이아웃 개선 사항

일부 Parallel NFS(pNFS) 서버가 더 나은 성능 통계를 수집할 수 있도록 하는 layoutstats() 작업을 제공합니다.

다음은 NFSv4.1의 기능입니다.

- 네트워크의 성능과 보안을 강화하며 pNFS에 대한 클라이언트 측 지원도 포함합니다.
- 더 이상 콜백에 대한 별도의 TCP 연결이 필요하지 않으므로, 예를 들어 NAT 또는 방화벽이 방해하는 경우도 NFS 서버에서 클라이언트에 연결할 수 없는 경우에도 NFS 서버에서 위임을 부여할 수 있습니다.
- 재부팅 작업을 제외하고 정확히 한 번만 의미 체계를 제공하므로, 일부 작업에서 응답이 손실되고 작업을 두 번 전송한 경우 특정 작업에서 부정확한 결과를 반환하는 경우가 있습니다.

4.3. NFSV3 및 NFSV4의 TCP 및 UDP 프로토콜

NFSv4에는 IP 네트워크를 통해 실행되는 TCP(Transmission Control Protocol)가 필요합니다.

NFSv3에서는 이전 Red Hat Enterprise Linux 버전에서 UDP(User Datagram Protocol)를 사용할 수도 있었습니다. Red Hat Enterprise Linux 9에서는 UDP를 통한 NFS가 더 이상 지원되지 않습니다. 기본적으로 UDP는 NFS 서버에서 비활성화됩니다.

4.4. NFS에 필요한 서비스

이 섹션에는 NFS 서버를 실행하거나 NFS 공유를 마운트하는 데 필요한 시스템 서비스가 나열되어 있습니다. Red Hat Enterprise Linux는 이러한 서비스를 자동으로 시작합니다.

Red Hat Enterprise Linux는 커널 수준 지원 및 서비스 프로세스를 결합하여 NFS 파일 공유를 제공합니다. 모든 NFS 버전은 클라이언트와 서버 간의 RPC(Remote Procedure calls)에 의존합니다. NFS 파일 시스템을 공유하거나 마운트하기 위해 다음 서비스가 구현되는 NFS 버전에 따라 함께 작동합니다.

nfsd

공유 NFS 파일 시스템에 대해 요청하는 NFS 서버 커널 모듈.

rpcbind

로컬 RPC 서비스의 포트 예약을 허용합니다. 그런 다음 이러한 포트를 사용할 수 있도록 (또는 알림) 해당 원격 RPC 서비스에 액세스할 수 있습니다. E gressIPbind 서비스는 RPC 서비스에 대한 요청에 응답하고 요청된 RPC 서비스에 대한 연결을 설정합니다. NFSv4에서는 사용되지 않습니다.

rpc.mountd

이 프로세스는 NFS 서버에서 NFSv3 클라이언트의 MOUNT 요청을 처리하는 데 사용됩니다. 요청된 NFS 공유를 현재 NFS 서버에서 내보내고 클라이언트가 액세스할 수 있는지 확인합니다. 마운트 요청이 허용되는 경우 nfs-mountd 서비스는 Success 상태로 응답하고 이 NFS 공유에 대한 File-Handle을 다시 NFS 클라이언트로 제공합니다.

rpc.nfsd

이 프로세스를 통해 서버가 광고하는 명시적인 NFS 버전 및 프로토콜을 사용할 수 있습니다. NFS 클라이언트가 연결할 때마다 서버 스레드 제공 등 NFS 클라이언트의 동적 요구를 충족하기 위해 Linux 커널과 함께 작동합니다. 이 프로세스는 nfs-server 서비스에 해당합니다.

lockd

이는 클라이언트와 서버에서 모두 실행되는 커널 스레드입니다. NFSv3 클라이언트가 서버의 파일을 잠글 수 있도록 하는 NLM(Network Lock Manager) 프로토콜을 구현합니다. NFS 서버를 실행하

고 NFS 파일 시스템이 마운트될 때마다 자동으로 시작됩니다.

rpc.statd

이 프로세스는 NFV(Network Status Monitor) RPC 프로토콜을 구현하여 NFS 서버가 정상적으로 종료되지 않고 다시 시작될 때 NFS 클라이언트에 알립니다. RuntimeClass-statd 서비스는 nfs-server 서비스에 의해 자동으로 시작되며 사용자 구성이 필요하지 않습니다. NFSv4에서는 사용되지 않습니다.

rpc.rquotad

이 프로세스는 원격 사용자에게 사용자 할당량 정보를 제공합니다. quota-rpc 패키지에서 제공하는 RuntimeClass-rquotad 서비스는 nfs-server 가 시작될 때 사용자가 시작해야 합니다.

rpc.idmapd

이 프로세스는 NFSv4 클라이언트 및 서버 백업 호출을 제공합니다. 이 호출은 사용자@도메인의 형식의 문자열과 로컬 UID 및 GID 사이에 매핑됩니다. idmapd 가 NFSv4에서 작동하려면 /etc/idmapd.conf 파일을 구성해야 합니다. 최소한 NFSv4 매핑 도메인을 정의하는 Domain 매개 변수를 지정해야 합니다. NFSv4 매핑 도메인이 DNS 도메인 이름과 동일한 경우 이 매개 변수를 건너뛸 수 있습니다. ID 매핑이 제대로 작동하려면 클라이언트 및 서버는 NFSv4 매핑 도메인에 동의해야 합니다.

NFSv4 서버만 nfs- idmapd 서비스에 의해 시작되는 RuntimeClass. idmapd를 사용합니다. NFSv4 클라이언트는 ID 매핑을 수행하기 위해 커널 온 디맨드에서 호출하는 인증 키 기반 nfsidmap 유틸리티를 사용합니다. nfsidmap 에 문제가 있는 경우 클라이언트는 Jenkinsfile .idmapd를 다시 사용합니다.

NFSv4를 사용한 RPC 서비스

NFSv4 프로토콜의 마운트 및 잠금 프로토콜이 통합되었습니다. 서버는 또한 잘 알려진 TCP 포트 2049에서 수신 대기합니다. 따라서 NFSv4는 EgressIP bind,lockd, RuntimeClass -statd 서비스와 상호 작용할 필요가 없습니다. nfs-mountd 서비스는 내보내기를 설정하기 위해 NFS 서버에 계속 필요하지만 over-the-wire 작업에는 포함되지 않습니다.

추가 리소스

- Diffie bind를 사용하지 않고 NFSv4만 구성.

4.5. NFS 호스트 이름 형식

이 섹션에서는 NFS 공유를 마운트하거나 내보낼 때 호스트를 지정하는 데 사용할 수 있는 다양한 형식을 설명합니다.

다음 형식으로 호스트를 지정할 수 있습니다.

단일 머신

다음 중 하나를 수행합니다.

- 정규화된 도메인 이름(서버에서 확인할 수 있음)
- 호스트 이름(서버에서 확인할 수 있음)
- IP 주소.

IP 네트워크

다음 형식 중 하나가 유효합니다.

- a.b.c.d /z.b.c.d는 네트워크이고 z 는 넷마스크의 비트 수입니다(예:DNS).
- a.b.c.d /netmask.b.c.d는 네트워크 및 넷마스크 입니다(예: 192.168.100.8/255.255.255.0).

netgroups

@group-name 형식 . 여기서 group-name 은 NIS netgroup 이름입니다.

4.6. NFS 서버 구성

이 섹션에서는 NFS 서버에서 내보내기를 구성하는 두 가지 방법의 구문 및 옵션에 대해 설명합니다.

- 수동으로 /etc/exports 구성 파일 편집
- 명령줄에서 exportfs 유틸리티 사용

4.6.1. /etc/exports 구성 파일

/etc/exports 파일은 원격 호스트로 내보낼 파일 시스템을 제어하고 옵션을 지정합니다. 다음 구문 규칙을 따릅니다.

- 빈 줄은 무시됩니다.
- 주석을 추가하려면 해시 표시(#)로 행을 시작합니다.
- 긴 줄을 백슬래시(\)로 래핑할 수 있습니다.
- 내보낸 각 파일 시스템은 고유한 개별 행에 있어야 합니다.
- 내보낸 파일 시스템 이후에 배치된 인증된 호스트 목록은 공백 문자로 구분해야 합니다.
- 호스트와 첫 번째 괄호를 분리하는 공백 없이 호스트 식별자 뒤에 직접 각 호스트에 대한 옵션을 배치해야 합니다.

내보내기 항목

내보낸 파일 시스템의 각 항목에는 다음과 같은 구조가 있습니다.

```
export host(options)
```

각 호스트에 대한 특정 옵션과 함께 여러 호스트를 지정할 수도 있습니다. 이렇게 하려면 공백으로 구분된 목록과 동일한 줄에 나열한 다음 각 호스트 이름 뒤에 해당 옵션(괄호 안에 있음)을 다음과 같이 나열합니다.

```
export host1(options1) host2(options2) host3(options3)
```

이 구조에서:

export

내보낼 디렉터리입니다.

호스트

공유되고 있는 호스트 또는 네트워크

옵션

호스트에 사용할 옵션

> ***예 4.1. 간단한 /etc/exports 파일***
>
> *가장 간단한 형식의 **/etc/exports** 파일은 내보낸 디렉터리와 액세스할 수 있는 호스트만 지정합니다.*
>
> ```
> /exported/directory bob.example.com
> ```
>
> *여기에서 **bob.example.com** 은 **NFS** 서버에서 **/exported/directory/** 를 마운트할 수 있습니다. 이 예제에 옵션이 지정되지 않기 때문에 **NFS**에서는 기본 옵션을 사용합니다.*

중요

/etc/exports 파일의 형식은 특히 공백 문자를 사용하는 것과 관련하여 매우 정확합니다. 내보낸 파일 시스템을 항상 호스트 및 호스트에서 공백 문자를 사용하여 서로 분리해야 합니다. 그러나 주석 행을 제외하고 파일에 다른 공백 문자는 없어야 합니다.

예를 들어 다음 두 줄은 동일한 것을 의미하는 것은 아닙니다.

```
/home bob.example.com(rw)
/home bob.example.com (rw)
```

*첫 번째 줄에서는 **bob.example.com** 의 사용자만 **/home** 디렉토리에 대한 읽기 및 쓰기 액세스 권한을 허용합니다. 두 번째 줄에서는 **bob.example.com** 의 사용자가 디렉터리를 읽기 전용(기본값)으로 마운트할 수 있는 반면 나머지 줄은 읽기/쓰기로 마운트할 수 있습니다.*

기본 옵션

내보내기 항목의 기본 옵션은 다음과 같습니다.

ro

내보낸 파일 시스템은 읽기 전용입니다. 원격 호스트는 파일 시스템에서 공유되는 데이터를 변경할 수 없습니다. 호스트가 파일 시스템(즉, 읽기 및 쓰기)을 변경할 수 있도록 하려면 **rw** 옵션을 지정합니다.

sync

이전 요청으로 변경한 내용이 디스크에 기록되기 전에 **NFS** 서버는 요청에 응답하지 않습니다. 대신 비동기 쓰기를 활성화하려면 **async** 옵션을 지정합니다.

wdelay

NFS 서버는 다른 쓰기 요청이 잘못된 것으로 의심되는 경우 디스크에 쓰는 것을 지연합니다. 이렇게 하면 별도의 쓰기 명령으로 디스크에 액세스해야 하는 횟수가 줄어들어 쓰기 오버헤드가 줄어들어 성능이 향상될 수 있습니다. 이 옵션을 비활성화하려면 기본 동기화 옵션도 지정된 경우에만 사용할 수 있는 **no_wdelay** 옵션을 지정합니다.

root_squash

이렇게 하면 **root** 사용자가 (로컬와 달리) **root** 사용자가 **root** 권한을 보유하지 못하도록 합니다. 대신 **NFS** 서버는 사용자 **ID nobody** 를 할당합니다. 이를 통해 원격 **root** 사용자의 기능을 가장 낮은 로컬 사용자에게 "스쿼시"하여 원격 서버에서 무단 쓰기를 방지합니다. 루트 스쿼시를 비활성화하려면 **no_root_squash** 옵션을 지정합니다.

모든 원격 사용자(**root** 포함)를 스쿼시하려면 **all_squash** 옵션을 사용합니다. **NFS** 서버에서 특정 호스트의 원격 사용자에게 할당해야 하는 사용자 및 그룹 **ID**를 지정하려면 다음과 같이 **anonuid** 및 **anongid** 옵션을 각각 사용합니다.

```
export host(anonuid=uid,anongid=gid)
```

여기에서 **uid** 및 **gid**는 사용자 **ID** 번호와 그룹 **ID** 번호입니다. **anonuid** 및 **anongid** 옵션을 사용하면 원격 **NFS** 사용자가 공유할 수 있는 특수 사용자 및 그룹 계정을 만들 수 있습니다.

기본적으로 **ACL**(액세스 제어 목록)은 **Red Hat Enterprise Linux**에서 **NFS**에서 지원됩니다. 이 기능을 비활성화하려면 파일 시스템을 내보낼 때 **no_acl** 옵션을 지정합니다.

기본값 및 재정의 옵션

내보낸 모든 파일 시스템의 각 기본값은 명시적으로 재정의해야 합니다. 예를 들어 **rw** 옵션이 지정되지 않으면 내보낸 파일 시스템이 읽기 전용으로 공유됩니다. 다음은 두 가지 기본 옵션을 재정의하는 **/etc/exports** 의 샘플 행입니다.

```
/another/exported/directory 192.168.0.3(rw,async)
```

이 예에서 **192.168.0.3** 은 **/another/exported/directory/ read** 및 **write**를 마운트할 수 있으며 디스크

에 대한 모든 쓰기는 비동기적입니다.

4.6.2. exportfs 유틸리티

exportfs 유틸리티를 사용하면 루트 사용자는 **NFS** 서비스를 다시 시작하지 않고 디렉터리를 선택적으로 내보내거나 내보내기 해제할 수 있습니다. 적절한 옵션이 제공되는 경우 **exportfs** 유틸리티는 내보낸 파일 시스템을 **/var/lib/nfs/xtab** 에 씁니다. **nfs-mountd** 서비스는 파일 시스템에 대한 액세스 권한을 결정할 때 **xtab** 파일을 참조하므로 내보낸 파일 시스템 목록에 대한 변경 사항이 즉시 적용됩니다.

공통 **exportfs** 옵션

다음은 **exportfs** 에 사용할 수 있는 일반적으로 사용되는 옵션 목록입니다.

-r

/var/lib/nfs/etab 에서 새 내보내기 목록을 구성하여 **/etc/exports** 에 나열된 모든 디렉터리를 내보낼 수 있습니다. 이 옵션은 **/etc/exports** 에 대한 변경 사항을 사용하여 내보내기 목록을 효과적으로 새로 고칩니다.

-a

exportfs 에 전달되는 다른 옵션에 따라 모든 디렉터리를 내보내거나 내보내지도록 합니다. 다른 옵션이 지정되지 않은 경우 **exportfs** 는 **/etc/exports** 에 지정된 모든 파일 시스템을 내보냅니다.

-o file-systems

/etc/exports 에 나열되지 않은 디렉토리를 지정합니다. 내보낼 추가 파일 시스템으로 파일 시스템을 교체합니다. 이러한 파일 시스템은 **/etc/exports** 에 지정된 것과 동일한 방식으로 포맷해야 합니다. 이 옵션은 내보낸 파일 시스템 목록에 영구적으로 추가하기 전에 내보낸 파일 시스템을 테스트하는 데 자주 사용됩니다.

-i

/etc/exports 를 무시합니다. 명령행에 제공된 옵션만 내보낸 파일 시스템을 정의하는 데 사용됩니다.

-u

Unexports 모든 공유 디렉터리를 해제합니다. **exportfs -ua** 명령은 모든 **NFS** 서비스를 유지하는 동안 **NFS** 파일 공유를 일시 중지합니다. **NFS** 공유를 다시 활성화하려면 **exportfs -r** 을 사용합니다.

-v

exportfs 명령이 실행될 때 내보내기 또는 내보내기 해제 중인 파일 시스템이 더 자세히 표시되는 자세한 작업입니다.

exportfs 유틸리티로 옵션이 전달되지 않으면 현재 내보낸 파일 시스템 목록이 표시됩니다.

추가 리소스

- NFS 호스트 이름 형식.

4.7. NFS 및 EGRESSIPBIND

E gressIPbind 서비스는 RPC(Remote Procedure call) 서비스를 수신 대기하는 포트에 매핑합니다. RPC 프로세스는 사용자가 수신 대기 중인 포트를 등록하고, 사용할 수 있는 RPC 프로그램 번호를 등록하기 시작할 때 EgressIP bind 에 알립니다. 그런 다음 클라이언트 시스템이 server에서 특정 RPC 프로그램 번호로 bind 에 연결합니다. controlPlane bind 서비스는 클라이언트를 적절한 포트 번호로 리디렉션하여 요청된 서비스와 통신할 수 있도록 합니다.

네트워크 파일 시스템 버전 3(NFSv3)에는 rpcbind 서비스가 필요합니다.

RPC 기반 서비스는verify bind 를 사용하여 들어오는 클라이언트 요청과의 모든 연결을 수행하므로 이러한 서비스를 시작하기 전에 EgressIP bind 를 사용할 수 있어야 합니다.

EgressIP bind에 대한 액세스 제어 규칙은 모든 RPC 기반 서비스에 영향을 미칩니다. 또는 NFS RPC 데몬마다 액세스 제어 규칙을 지정할 수 있습니다.

추가 리소스

- rpc.mountd(8) man page
- RPC.statd(8) 도움말 페이지

4.8. NFS 설치

이 절차에서는 NFS 공유를 마운트하거나 내보내는 데 필요한 모든 패키지를 설치합니다.

절차

- nfs-utils 패키지를 설치합니다.

```
# dnf install nfs-utils
```

4.9. NFS 서버 시작

다음 절차에서는 NFS 공유를 내보내는 데 필요한 NFS 서버를 시작하는 방법을 설명합니다.

사전 요구 사항

- NFSv3 연결을 지원하는 서버의 경우 N etNamespacebind 서비스가 실행되고 있어야 합니다. EgressIP bind 가 활성화되어 있는지 확인하려면 다음 명령을 사용합니다.

```
$ systemctl status rpcbind
```

서비스가 중지된 경우 이를 시작하고 활성화합니다.

```
$ systemctl enable --now rpcbind
```

절차

- NFS 서버를 시작하고 부팅 시 자동으로 시작되도록 하려면 다음 명령을 사용합니다.

```
# systemctl enable --now nfs-server
```

추가 리소스

- NFSv4 전용 서버 구성.

4.10. NFS 및 EGRESSIPBIND 문제 해결

controlPlane bind 서비스는 RPC 서비스와 통신하는 데 사용되는 포트 번호 간의 조정을 제공하므로 문제를 해결할 때 stderr bind 를 사용하여 현재 RPC 서비스의 상태를 확인하는 것이 유용합니다. RuntimeClass info 유틸리티는 각 RPC 기반 서비스와 포트 번호, RPC 프로그램 번호, 버전 번호, IP 프로토콜 유형(TCP 또는 UDP)을 보여줍니다.

절차

1. EgressIP bind 에 대해 적절한 NFS RPC 기반 서비스가 활성화되어 있는지 확인하려면 다음 명령을 사용하십시오.

```
# rpcinfo -p
```

예 4.2. RuntimeClassinfo -p 명령 출력

다음은 이 명령의 출력 샘플입니다.

```
program vers proto   port  service
 100000    4   tcp    111  portmapper
 100000    3   tcp    111  portmapper
 100000    2   tcp    111  portmapper
 100000    4   udp    111  portmapper
 100000    3   udp    111  portmapper
 100000    2   udp    111  portmapper
 100005    1   udp  20048  mountd
 100005    1   tcp  20048  mountd
 100005    2   udp  20048  mountd
 100005    2   tcp  20048  mountd
 100005    3   udp  20048  mountd
 100005    3   tcp  20048  mountd
 100024    1   udp  37769  status
 100024    1   tcp  49349  status
 100003    3   tcp   2049  nfs
 100003    4   tcp   2049  nfs
 100227    3   tcp   2049  nfs_acl
 100021    1   udp  56691  nlockmgr
 100021    3   udp  56691  nlockmgr
 100021    4   udp  56691  nlockmgr
 100021    1   tcp  46193  nlockmgr
 100021    3   tcp  46193  nlockmgr
 100021    4   tcp  46193  nlockmgr
```

NFS 서비스 중 하나가 올바르게 시작되지 않으면 **E gressIPbind** 가 해당 서비스의 **RPC** 요청을 올바른 포트로 매핑할 수 없습니다.

2. 대부분의 경우 **NFS**가 **vGPU info** 출력에 없는 경우 **NFS**를 다시 시작하면 **NFS**를 다시 시작하면 서비스가 **EgressIP bind** 에 올바르게 등록되고 작동을 시작합니다.

```
# systemctl restart nfs-server
```

추가 리소스

- **NFSv4** 전용 서버 구성.

4.11. 방화벽 뒤에서 실행되도록 NFS 서버 구성

NFS에는 RPC 서비스에 대한 포트를 동적으로 할당하고 방화벽 규칙을 구성하는 데 문제가 발생할 수 있는 EgressIP bind 서비스가 필요합니다. 다음 섹션에서는 지원하려는 경우 방화벽에서 작동하도록 NFS 버전을 구성하는 방법을 설명합니다.

- NFSv3

  여기에는 NFSv3를 지원하는 모든 서버가 포함됩니다.

  - NFSv3- 전용 서버
  - NFSv3 및 NFSv4를 모두 지원하는 서버

- NFSv4 전용

4.11.1. 방화벽 뒤에서 실행되도록 NFSv3- 사용 서버 구성

다음 절차에서는 방화벽 뒤에서 실행되도록 NFSv3를 지원하는 서버를 구성하는 방법을 설명합니다. 여기에는 NFSv3 및 NFSv4를 모두 지원하는 NFSv3 전용 서버 및 서버가 포함됩니다.

절차

1. 클라이언트가 방화벽 뒤에서 NFS 공유에 액세스할 수 있도록 하려면 NFS 서버에서 다음 명령을 실행하여 방화벽을 구성합니다.

```
firewall-cmd --permanent --add-service mountd
firewall-cmd --permanent --add-service rpc-bind
firewall-cmd --permanent --add-service nfs
```

2. 다음과 같이 /etc/nfs.conf 파일에서 RPC 서비스 nlockmgr 에서 사용할 포트를 지정합니다.

```
[lockd]

port=tcp-port-number
udp-port=udp-port-number
```

   또는 /etc/modprobe.d/lockd.conf 파일에 nlm_tcpport 및 nlm_udpport 를 지정할 수 있습니다.

3. *NFS 서버에서 다음 명령을 실행하여 방화벽에서 지정된 포트를 엽니다.*

```
firewall-cmd --permanent --add-port=<lockd-tcp-port>/tcp
firewall-cmd --permanent --add-port=<lockd-udp-port>/udp
```

4. *다음과 같이 /etc/nfs . conf 파일의 [statd] 섹션을 편집하여 alice.statd에 대한 정적 포트를 추가합니다.*

```
[statd]

port=port-number
```

5. *NFS 서버에서 다음 명령을 실행하여 방화벽에서 추가된 포트를 엽니다.*

```
firewall-cmd --permanent --add-port=<statd-tcp-port>/tcp
firewall-cmd --permanent --add-port=<statd-udp-port>/udp
```

6. *방화벽 구성을 다시 로드합니다.*

```
firewall-cmd --reload
```

7. *먼저-02 --statd 서비스를 다시 시작한 다음 nfs-server 서비스를 다시 시작합니다.*

```
# systemctl restart rpc-statd.service
# systemctl restart nfs-server.service
```

또는 /etc/modprobe.d/ lockd.conf 파일에 lockd 포트를 지정한 경우:

a. */proc/sys/fs/nfs/nlm_tcpport 및 /proc/sys/fs/nfs/nlm_udpport 의 현재 값을 업데이트합니다.*

```
# sysctl -w fs.nfs.nlm_tcpport=<tcp-port>
# sysctl -w fs.nfs.nlm_udpport=<udp-port>
```

b. *-02- -statd 및 nfs-server 서비스를 다시 시작합니다.*

```
# systemctl restart rpc-statd.service
# systemctl restart nfs-server.service
```

4.11.2. 방화벽 뒤에서 실행되도록 NFSv4 전용 서버 구성

다음 절차에서는 방화벽 뒤에서 실행되도록 NFSv4 전용 서버를 구성하는 방법을 설명합니다.

절차

1. 클라이언트가 방화벽 뒤에서 NFS 공유에 액세스할 수 있도록 하려면 NFS 서버에서 다음 명령을 실행하여 방화벽을 구성합니다.

```
firewall-cmd --permanent --add-service nfs
```

2. 방화벽 구성을 다시 로드합니다.

```
firewall-cmd --reload
```

3. nfs-server를 다시 시작합니다.

```
# systemctl restart nfs-server
```

4.11.3. 방화벽 뒤에서 실행되도록 NFSv3 클라이언트 구성

방화벽 뒤에서 실행되도록 NFSv3 클라이언트를 구성하는 절차는 방화벽 뒤에서 실행되도록 NFSv3 서버를 구성하는 절차와 유사합니다.

구성 중인 시스템이 NFS 클라이언트와 NFS 서버 둘 다인 경우, NFSv3- 사용 서버 구성에 설명된 절차에 따라 방화벽 뒤에서 실행됩니다.

다음 절차에서는 방화벽 뒤에서 실행하기 위해서만 NFS 클라이언트인 시스템을 구성하는 방법을 설명합니다.

절차

1. 클라이언트가 방화벽 뒤에 있을 때 NFS 서버가 NFS 클라이언트에 콜백을 수행할 수 있도록 하려면 NFS 클라이언트에서 다음 명령을 실행하여 방화벽에 CHAP -bind 서비스를 추가합니다.

```
firewall-cmd --permanent --add-service rpc-bind
```

2. 다음과 같이 /etc/nfs.conf 파일에서 RPC 서비스 nlockmgr 에서 사용할 포트를 지정합니다.

```
[lockd]

port=port-number
udp-port=upd-port-number
```

또는 /etc/modprobe.d/lockd.conf 파일에 nlm_tcpport 및 nlm_udpport 를 지정할 수 있습니다.

3. NFS 클라이언트에서 다음 명령을 실행하여 방화벽에서 지정된 포트를 엽니다.

```
firewall-cmd --permanent --add-port=<lockd-tcp-port>/tcp
firewall-cmd --permanent --add-port=<lockd-udp-port>/udp
```

4. 다음과 같이 /etc/nfs . conf 파일의 [statd] 섹션을 편집하여 alice.statd에 대한 정적 포트를 추가합니다.

```
[statd]

port=port-number
```

5. NFS 클라이언트에서 다음 명령을 실행하여 방화벽에서 추가된 포트를 엽니다.

```
firewall-cmd --permanent --add-port=<statd-tcp-port>/tcp
firewall-cmd --permanent --add-port=<statd-udp-port>/udp
```

6. 방화벽 구성을 다시 로드합니다.

```
firewall-cmd --reload
```

7. -02- -statd 서비스를 다시 시작합니다.

```
# systemctl restart rpc-statd.service
```

또는 /etc/modprobe.d/ lockd.conf 파일에 lockd 포트를 지정한 경우:

a. /proc/sys/fs/nfs/nlm_tcpport 및 /proc/sys/fs/nfs/nlm_udpport 의 현재 값을 업데이트합니다.

```
# sysctl -w fs.nfs.nlm_tcpport=<tcp-port>
# sysctl -w fs.nfs.nlm_udpport=<udp-port>
```

b. -02- -statd 서비스를 다시 시작합니다.

```
# systemctl restart rpc-statd.service
```

4.11.4. 방화벽 뒤에서 실행되도록 NFSv4 클라이언트 구성

클라이언트가 NFSv4.0을 사용하는 경우에만 이 절차를 수행하십시오. 이 경우 NFSv4.0 콜백용 포트를 열어야 합니다.

이후 프로토콜 버전에서는 서버가 클라이언트에서 시작한 동일한 연결에서 콜백을 수행하기 때문에 NFSv4.1 이상에는 이 절차가 필요하지 않습니다.

절차

1. NFSv4.0 콜백이 방화벽을 통과하도록 허용하려면 /proc/sys/fs/nfs_callback_tcpport 를 설정하고 서버가 다음과 같이 클라이언트의 해당 포트에 연결할 수 있습니다.

```
# echo "fs.nfs.nfs_callback_tcpport = <callback-port>" >/etc/sysctl.d/90-nfs-callback-port.conf
# sysctl -p /etc/sysctl.d/90-nfs-callback-port.conf
```

2. NFS 클라이언트에서 다음 명령을 실행하여 방화벽에서 지정된 포트를 엽니다.

```
firewall-cmd --permanent --add-port=<callback-port>/tcp
```

3. 방화벽 구성을 다시 로드합니다.

```
firewall-cmd --reload
```

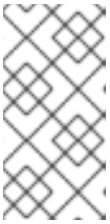
4.12. 방화벽을 통해 RPC 할당량 내보내기

디스크 할당량을 사용하는 파일 시스템을 내보내는 경우 **quota Remote Procedure call(RPC)** 서비스를 사용하여 **NFS** 클라이언트에 디스크 할당량 데이터를 제공할 수 있습니다.

절차

1. **RuntimeClass -rquotad** 서비스를 활성화하고 시작합니다.

```
# systemctl enable --now rpc-rquotad
```



참고

nfs -rquotad 서비스는 활성화된 경우 **nfs-server** 서비스를 시작한 후 자동으로 시작됩니다.

2. 방화벽 뒤에서 할당량 **RPC** 서비스에 액세스할 수 있도록 하려면 **TCP**(또는 **UDP**가 활성화된 경우 **UDP**) 포트 **875**를 열어야 합니다. 기본 포트 번호는 **/etc/services** 파일에 정의되어 있습니다.

/etc/sysconfig/rpc -rquotad 파일의 **RPCRQUOTADOPTS** 변수에 **-p** 포트 번호를 추가하여 기본 포트 번호를 재정의할 수 있습니다.

3. 기본적으로 원격 호스트는 할당량만 읽을 수 있습니다. 클라이언트가 할당량을 설정할 수 있도록 하려면 **/etc/sysconfig/rpc-rquotad** 파일의 **RPCRQUOTADOPTS** 변수에 **-S** 옵션을 추가합니다.

4. **/etc/sysconfig/ rpc-rquotad** 파일의 변경 사항을 적용하려면 **6443-rquotad**를 다시 시작합니다.

```
# systemctl restart rpc-rquotad
```

4.13. NFS OVER RDMA (NFSORDMA)

RDMA(Remote Direct Memory Access) 서비스는 **Red Hat Enterprise Linux 9**의 **RDMA** 가능 하드웨어에서 자동으로 작동합니다.

절차

1. rdma-core 패키지를 설치합니다.

```
# dnf install rdma-core
```

2. /etc/rdma/modules/rdma.conf 파일에서 xprtrdma 및 svcrdma 가 있는 행을 주석 처리했는지 확인합니다.

```
# NFS over RDMA client support
xprtrdma
# NFS over RDMA server support
svcrdma
```

3. NFS 서버에서 디렉토리 /mnt/nfsordma 를 만들고 /etc/exports 로 내보냅니다.

```
# mkdir /mnt/nfsordma
# echo "/mnt/nfsordma *(fsid=0,rw,async,insecure,no_root_squash)" >> /etc/exports
```

4. NFS 클라이언트에서 서버 IP 주소(예: 172.31.0.186)를 사용하여 nfs-share를 마운트합니다.

```
# mount -o rdma,port=20049 172.31.0.186:/mnt/nfs-share /mnt/nfs
```

5. nfs-server 서비스를 다시 시작합니다.

```
# systemctl restart nfs-server
```

추가 리소스

- RFC 5667 표준

4.14. 추가 리소스

- Linux NFS why

5장. NFSV4 전용 서버 구성

NFS 서버 관리자는 시스템에서 열려 있는 포트 및 실행 중인 서비스 수를 최소화하는 NFSv4만 지원하도록 NFS 서버를 구성할 수 있습니다.

5.1. NFSV4 전용 서버의 이점 및 단점

이 섹션에서는 NFSv4만 지원하도록 NFS 서버를 구성하는 이점과 단점에 대해 설명합니다.

기본적으로 NFS 서버는 Red Hat Enterprise Linux 9에서 NFSv3 및 NFSv4 연결을 지원합니다. 그러나 NFS 버전 4.0 이상만 지원하도록 NFS를 구성할 수도 있습니다. 이렇게 하면 NFSv4가 네트워크에서 수신 대기할 필요가 없으므로 열린 포트 수와 시스템에서 실행 중인 서비스 수가 최소화됩니다.

NFS 서버가 NFSv4 전용으로 구성된 경우 NFSv3를 사용하여 공유를 마운트하려고 하는 클라이언트는 다음과 같은 오류와 함께 실패합니다.

```
Requested NFS version or transport protocol is not supported.
```

선택적으로 NFSv4 전용 사례에 필요하지 않은 RPCBIND,MOUNT, NSM 프로토콜 호출에 대한 수신을 비활성화할 수도 있습니다.

이러한 추가 옵션을 비활성화하는 영향은 다음과 같습니다.

- NFSv3를 사용하여 서버의 공유를 마운트하려는 클라이언트가 응답하지 않습니다.
- NFS 서버 자체는 NFSv3 파일 시스템을 마운트할 수 없습니다.

5.2. NFSV4만 지원하도록 NFS 서버 구성

다음 절차에서는 NFS 버전 4.0 이상만 지원하도록 NFS 서버를 구성하는 방법을 설명합니다.

절차

1. /etc/nfs.conf 구성 파일의 [nfsd] 섹션에 다음 행을 추가하여 NFSv3를 비활성화합니다.

```
[nfsd]

vers3=no
```

2. **NFSv4** 전용 경우에는 필요하지 않은 **RPCBIND,MOUNT, NSM** 프로토콜 호출 수신 대기를 비활성화합니다. 관련 서비스를 비활성화합니다.

```
# systemctl mask --now rpc-statd.service rpcbind.service rpcbind.socket
```

3. **NFS** 서버를 다시 시작합니다.

```
# systemctl restart nfs-server
```

변경 사항은 **NFS** 서버를 시작하거나 다시 시작하는 즉시 적용됩니다.

5.3. NFSV4 전용 구성 확인

이 절차에서는 **netstat** 유틸리티를 사용하여 **NFS** 서버가 **NFSv4** 전용 모드에서 구성되었는지 확인하는 방법을 설명합니다.

절차

- **netstat** 유틸리티를 사용하여 **TCP** 및 **UDP** 프로토콜에서 수신 대기하는 서비스를 나열합니다.

```
# netstat --listening --tcp --udp
```

예 5.1. NFSv4 전용 서버의 출력

다음은 **NFSv4** 전용 서버의 **netstat** 출력 예입니다. **RPCBIND,MOUNT, NSM** 의 수신 대기도 비활성화됩니다. 여기서 **nfs** 는 유일하게 수신 대기하는 **NFS** 서비스입니다.

```
# netstat --listening --tcp --udp

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:nfs             0.0.0.0:*               LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
tcp6       0      0 [::]:nfs                [::]:*                  LISTEN
udp        0      0 localhost.locald:bootpc 0.0.0.0:*
```

예 5.2. NFSv4 전용 서버를 구성하기 전에 출력

비교에서 **NFSv4** 전용 서버를 구성하기 전에 **netstat** 출력에 **Sunrpc** 및 **mountd** 서비스가 포함됩니다.

```
# netstat --listening --tcp --udp

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address State
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*       LISTEN
tcp        0      0 0.0.0.0:40189           0.0.0.0:*       LISTEN
tcp        0      0 0.0.0.0:46813           0.0.0.0:*       LISTEN
tcp        0      0 0.0.0.0:nfs             0.0.0.0:*       LISTEN
tcp        0      0 0.0.0.0:sunrpc          0.0.0.0:*       LISTEN
tcp        0      0 0.0.0.0:mountd          0.0.0.0:*       LISTEN
tcp6       0      0 [::]:ssh                [::]:*          LISTEN
tcp6       0      0 [::]:51227              [::]:*          LISTEN
tcp6       0      0 [::]:nfs                [::]:*          LISTEN
tcp6       0      0 [::]:sunrpc             [::]:*          LISTEN
tcp6       0      0 [::]:mountd             [::]:*          LISTEN
tcp6       0      0 [::]:45043              [::]:*          LISTEN
udp        0      0 localhost:1018          0.0.0.0:*
udp        0      0 localhost.locald:bootpc 0.0.0.0:*
udp        0      0 0.0.0.0:mountd          0.0.0.0:*
udp        0      0 0.0.0.0:46672           0.0.0.0:*
udp        0      0 0.0.0.0:sunrpc          0.0.0.0:*
udp        0      0 0.0.0.0:33494           0.0.0.0:*
udp6       0      0 [::]:33734              [::]:*
udp6       0      0 [::]:mountd             [::]:*
udp6       0      0 [::]:sunrpc             [::]:*
udp6       0      0 [::]:40243              [::]:*
```

6장. NFS 보안

NFS 보안 위험을 최소화하고 서버에서 데이터를 보호하려면 서버에서 NFS 파일 시스템을 내보내거나 클라이언트에서 데이터를 마운트할 때 다음 섹션을 고려하십시오.

6.1. AUTH_SYS 및 내보내기 제어를 사용한 NFS 보안

NFS는 내보낸 파일에 대한 액세스를 제어하기 위해 다음과 같은 기존 옵션을 제공합니다.

- 서버는 IP 주소 또는 호스트 이름으로 파일 시스템을 마운트할 수 있는 호스트를 제한합니다.
- 서버는 로컬 사용자와 동일한 방식으로 NFS 클라이언트의 사용자에게 파일 시스템 권한을 적용합니다. 일반적으로 NFS는 AUTH_SYS 호출 메시지(AUTH_UNIX라고도 함)를 사용하여 이 작업을 수행합니다. 이 메시지는 클라이언트에 사용자의 UID 및 GID를 나타냅니다. 이는 악의적인 또는 잘못 구성된 클라이언트가 이 오류를 쉽게 가져오고 사용자가 그렇지 않아야 하는 파일에 액세스할 수 있도록 할 수 있음을 의미합니다.

잠재적인 위험을 제한하기 위해 관리자는 종종 액세스를 읽기 전용 또는 squash 사용자 권한으로 일반 사용자 및 그룹 ID로 제한합니다. 유감스럽게도 이러한 솔루션은 NFS 공유를 원래 의도한 방식으로 사용하지 않습니다.

또한 공격자가 NFS 파일 시스템을 내보내는 시스템에서 사용하는 DNS 서버를 제어하면 특정 호스트 이름 또는 정규화된 도메인 이름과 연결된 시스템을 인증되지 않은 시스템으로 가리킬 수 있습니다. 이 시점에 는 NFS 마운트에 대한 추가 보안을 제공하기 위해 사용자 이름 또는 암호 정보가 교환되지 않으므로 인증되지 않은 시스템은 NFS 공유를 마운트할 수 있는 시스템입니다.

와일드카드는 NFS를 통해 디렉터리를 내보낼 때는 의도한 것보다 많은 시스템을 포함할 수 있으므로 와일드카드를 사용해야 합니다.

추가 리소스

- NFS 및 EgressIP bind 를 보호하려면 예를 들어 nftables 및 firewalld 를 사용합니다.
- nft(8) 매뉴얼 페이지

- *firewalld-cmd(1) 매뉴얼 페이지*

6.2. AUTH_GSS를 사용한 NFS 보안

모든 버전의 NFS는 RPCSEC_GSS 및 Kerberos 메커니즘을 지원합니다.

RPCSEC_GSS Kerberos 메커니즘을 사용하는 AUTH_SYS와 달리 서버는 파일에 액세스하는 사용자를 올바르게 나타내기 위해 클라이언트에 의존하지 않습니다. 대신 암호화는 서버에 사용자를 인증하는 데 사용되며 악의적인 클라이언트가 해당 사용자의 Kerberos 자격 증명을 사용하지 않고 가장할 수 있습니다. Kerberos를 구성한 후 추가 설정이 필요 없기 때문에 RPCSEC_GSS Kerberos 메커니즘을 사용하는 가장 간단한 방법은 마운트를 보호하는 가장 간단한 방법입니다.

6.3. KERBEROS를 사용하도록 NFS 서버 및 클라이언트 구성

Kerberos는 대칭 암호화 및 신뢰할 수 있는 타사인 NetNamespace를 사용하여 클라이언트와 서버가 서로 인증할 수 있도록 하는 네트워크 인증 시스템입니다. Kerberos 설정을 위해 IdM(Identity Management)을 사용하는 것이 좋습니다.

사전 요구 사항

- Kerberos 키 배포 센터(mtls)가 설치 및 구성되어 있습니다.

절차

1.
   - NFS 서버 측에서 nfs/hostname.domain@REALM principal를 생성합니다.
   - 서버와 클라이언트 쪽 모두에 host/hostname.domain@REALM principal를 생성합니다.
   - 클라이언트 및 서버의 키 탭에 해당 키를 추가합니다.

2. 서버 측에서 sec= 옵션을 사용하여 원하는 보안 플레이버를 활성화합니다. 모든 보안 플레이버 및 암호가 아닌 마운트를 활성화하려면 다음을 수행합니다.

```
/export *(sec=sys:krb5:krb5i:krb5p)
```

SEC= 옵션과 함께 사용할 유효한 보안 플레이버는 다음과 같습니다.

- **sys: no cryptographic protection, the default**
- **RuntimeClass5:** 인증 전용
- **RuntimeClass5i:** 무결성 보호
  - 사용자 인증에 **Kerberos V5**를 사용하고 데이터 변조를 방지하기 위해 보안 체크섬을 사용하여 **NFS** 작업의 무결성 검사를 수행합니다.
- **RuntimeClass5p:** 개인 정보 보호
  - **Kerberos V5**를 사용하여 사용자 인증, 무결성 검사 및 **NFS** 트래픽을 암호화하여 트래픽 스니핑을 방지합니다. 이는 가장 안전한 설정이지만 성능 오버헤드가 가장 많습니다.

3. 클라이언트 측에서 **sec=krb5 i(**또는 **setup**에 따라 초 **=krb5i, 초보5p)**를 마운트 옵션에 추가합니다.

```
# mount -o sec=krb5 server:/export /mnt
```

추가 리소스

- **krb5** 보안 **NFS**에서 **root**로 파일 만들기**.** 권장되지 않음**.**
- **export(5)** 도움말 페이지
- **NFS(5)** 도움말 페이지

6.4. NFSV4 보안 옵션

NFSv4에는 Microsoft Windows NT 모델의 기능 및 광범위한 배포 때문에 POSIX 모델이 아닌 Microsoft Windows NT 모델을 기반으로 하는 ACL 지원이 포함되어 있습니다.

NFSv4의 또 다른 중요한 보안 기능은 MOUNT 프로토콜 사용을 제거하여 파일 시스템 마운트입니다. MOUNT 프로토콜에는 프로토콜이 파일을 처리하는 방식 때문에 보안 위험이 있었습니다.

6.5. 마운트된 NFS 내보내기에 대한 파일 권한

NFS 파일 시스템이 원격 호스트에서 읽기 또는 읽기 및 쓰기로 마운트되면 각 공유 파일에 대한 유일한 보호 권한이 있습니다. 동일한 사용자 ID 값을 공유하는 두 사용자가 서로 다른 클라이언트 시스템에 동일한 NFS 파일 시스템을 마운트한 경우 서로의 파일을 수정할 수 있습니다. 또한 클라이언트 시스템에서 root로 로그인한 사용자는 su - 명령을 사용하여 NFS 공유로 모든 파일에 액세스할 수 있습니다.

기본적으로 ACL(액세스 제어 목록)은 Red Hat Enterprise Linux에서 NFS에서 지원됩니다. Red Hat 은 이 기능을 활성화하는 것을 권장합니다.

기본적으로 NFS는 파일 시스템을 내보낼 때 루트 스쿼딩 을 사용합니다. 이렇게 하면 로컬 시스템에서 root 사용자로 NFS 공유에 액세스하는 사용자의 사용자 ID를 nobody 로 설정합니다. 루트 스쿼시는 기본 옵션 root_squash 에 의해 제어됩니다. 이 옵션에 대한 자세한 내용은 NFS 서버 구성을 참조하십시오.

NFS 공유를 읽기 전용으로 내보낼 때는 all_squash 옵션을 사용하는 것이 좋습니다. 이 옵션을 사용하면 내보낸 파일 시스템에 액세스하는 모든 사용자가 nobody 사용자의 사용자 ID를 사용합니다.

7장. NFS에서 PNFS SCSI 레이아웃 활성화

pNFS SCSI은 데이터에 액세스하는 데 pNFS SCSI 레이아웃을 사용하도록 NFS 서버와 클라이언트를 구성할 수 있습니다. pNFS SCSI은 파일에 대한 장기간 단일 클라이언트 액세스 권한이 필요한 사용 사례에서 유용합니다.

사전 요구 사항

- 클라이언트와 서버는 모두 동일한 블록 장치에 SCSI 명령을 보낼 수 있어야 합니다. 즉, 블록 장치는 공유 SCSI 버스에 있어야 합니다.
- 블록 장치에는 XFS 파일 시스템이 포함되어야 합니다.
- SCSI 장치는 SCSI-3 기본 명령 사양에 설명된 대로 SCSI 영구 예약을 지원해야 합니다.

7.1. PNFS 기술

pNFS 아키텍처는 NFS의 확장성을 향상시킵니다. 서버가 pNFS를 구현하는 경우 클라이언트는 여러 서버를 통해 동시에 데이터에 액세스할 수 있습니다. 이로 인해 성능이 개선될 수 있습니다.

pNFS는 RHEL에서 다음과 같은 스토리지 프로토콜 또는 레이아웃을 지원합니다.

- 파일
- flexfiles
- SCSI

7.2. PNFS SCSI 레이아웃

SCSI 레이아웃은 pNFS 블록 레이아웃 작업을 기반으로 합니다. 레이아웃은 SCSI 장치 간에 정의됩니다. SCSI 영구 예약을 지원할 수 있어야 하는 논리 단위(LU)로 순차적 일련의 고정 크기 블록이 포함되어 있습니다. LU 장치는 SCSI 장치 식별으로 식별됩니다.

pNFS SCSI은 파일에 대한 장기간 단일 클라이언트 액세스와 관련된 사용 사례에서 잘 수행합니다. 예를 들어 메일 서버 또는 클러스터를 보유하는 가상 시스템일 수 있습니다.

클라이언트와 서버 간 작업

NFS 클라이언트가 파일에서 읽거나 쓸 때 클라이언트는 LAYOUTGET 작업을 수행합니다. 서버는 SCSI 장치에 있는 파일의 위치로 응답합니다. 클라이언트는 사용할 SCSI 장치를 결정하기 위해 GETDEVICEINFO 의 추가 작업을 수행해야 할 수 있습니다. 이러한 작업이 올바르게 작동하는 경우 클라이언트는 READ 및 WRITE 작업을 서버로 보내는 대신 SCSI 장치에 직접 I/O 요청을 실행할 수 있습니다.

클라이언트 간의 오류 또는 경합으로 인해 서버에서 레이아웃을 회수하거나 클라이언트에 문제를 발생시키지 않을 수 있습니다. 이러한 경우 클라이언트는 I/O 요청을 SCSI 장치로 직접 보내는 대신 READ 및 WRITE 작업을 서버에 다시 실행합니다.

작업을 모니터링하려면 pNFS SCSI 레이아웃 기능 모니터링 을 참조하십시오.

장치 예약

PNFS SCSI은 예약 할당을 통해 펜싱을 처리합니다. 서버가 클라이언트에 레이아웃을 생성하기 전에 등록된 클라이언트만 장치에 액세스할 수 있도록 SCSI 장치를 예약합니다. 클라이언트가 해당 SCSI 장치에 명령을 실행할 수 있지만 장치에 등록되지 않은 경우 해당 장치의 클라이언트의 많은 작업이 실패합니다. 예를 들어, 서버에 클라이언트에 해당 장치의 레이아웃이 제공되지 않은 경우 클라이언트의 blkid 명령은 XFS 파일 시스템의 UUID를 표시하지 않습니다.

서버가 자체 영구 예약을 제거하지 않습니다. 이렇게 하면 클라이언트 및 서버를 다시 시작해도 장치에서 파일 시스템 내의 데이터가 보호됩니다. SCSI 장치를 다시 사용하려면 NFS 서버에서 영구 예약을 수동으로 제거해야 할 수 있습니다.

7.3. PNFS와 호환되는 SCSI 장치 확인

다음 절차에서는 SCSI 장치가 pNFS SCSI 레이아웃을 지원하는지 확인합니다.

사전 요구 사항

- sg3_utils 패키지를 설치합니다.

```
# dnf install sg3_utils
```

절차

- 서버와 클라이언트 모두에서 적절한 **SCSI** 장치 지원이 있는지 확인합니다.

```
# sg_persist --in --report-capabilities --verbose path-to-scsi-device
```

Persist throughput Power Loss Active (PTPL_A) bit가 설정되어 있는지 확인합니다.

예 7.1. pNFS SCSI을 지원하는 **SCSI** 장치

다음은 **pNFS SCSI**을 지원하는 **SCSI** 장치의 **sg_persist** 출력의 예입니다. **PTPL_A** 비트는 **1** 을 보고합니다.

```
  inquiry cdb: 12 00 00 00 24 00
  Persistent Reservation In cmd: 5e 02 00 00 00 00 00 20 00 00
LIO-ORG   block11          4.0
Peripheral device type: disk
Report capabilities response:
 Compatible Reservation Handling(CRH): 1
 Specify Initiator Ports Capable(SIP_C): 1
 All Target Ports Capable(ATP_C): 1
 Persist Through Power Loss Capable(PTPL_C): 1
 Type Mask Valid(TMV): 1
 Allow Commands: 1
 Persist Through Power Loss Active(PTPL_A): 1
  Support indicated in Type mask:
   Write Exclusive, all registrants: 1
   Exclusive Access, registrants only: 1
   Write Exclusive, registrants only: 1
   Exclusive Access: 1
   Write Exclusive: 1
   Exclusive Access, all registrants: 1
```

추가 리소스

- **sg_persist(8)** 매뉴얼 페이지

7.4. 서버에서 PNFS SCSI 설정

이 절차에서는 **pNFS SCSI** 레이아웃을 내보내도록 **NFS** 서버를 구성합니다.

절차

1. 서버에서 SCSI 장치에 생성된 XFS 파일 시스템을 마운트합니다.

2. NFS 버전 4.1 이상을 내보내도록 NFS 서버를 구성합니다. /etc/nfs.conf 파일의 [nfsd] 섹션에 다음 옵션을 설정합니다.

```
[nfsd]

vers4.1=y
```

3. pnfs 옵션을 사용하여 NFS를 통해 XFS 파일 시스템을 내보내도록 NFS 서버를 구성합니다.

예 7.2. pNFS SCSI 내보내기를 위한 /etc/exports 항목

/etc/exports 구성 파일의 다음 항목에서는 /exported/directory/ 에 마운트된 파일 시스템을 allowed.example.com 클라이언트에 pNFS SCSI 레이아웃으로 내보냅니다.

```
/exported/directory allowed.example.com(pnfs)
```

참고

내보낸 파일 시스템은 파티션뿐만 아니라 전체 블록 장치에 생성해야 합니다.

추가 리소스

- NFS 공유 내보내기.

7.5. 클라이언트에서 PNFS SCSI 설정

이 절차에서는 NFS 클라이언트가 pNFS SCSI 레이아웃을 마운트하도록 구성합니다.

사전 요구 사항

- NFS 서버는 pNFS SCSI을 통해 XFS 파일 시스템을 내보내도록 구성되어 있습니다. 서버에서 pNFS SCSI 설정을 참조하십시오.

절차

- 클라이언트에서 **NFS** 버전 **4.1** 이상을 사용하여 내보낸 **XFS** 파일 시스템을 마운트합니다.

```
# mount -t nfs -o nfsvers=4.1 host:/remote/export /local/directory
```

NFS 없이 **XFS** 파일 시스템을 직접 마운트하지 마십시오.

추가 리소스

- *NFS 공유* 마운트 .

7.6. 서버에서 PNFS SCSI 예약 해제

이 절차에서는 **NFS** 서버가 **SCSI** 장치에 보유하고 있는 영구 예약을 해제합니다. 이를 통해 더 이상 **pNFS SCSI**을 내보낼 필요가 없는 경우 **SCSI** 장치를 변경할 수 있습니다.

서버에서 예약을 제거해야 합니다. 다른 **IT Nexus**에서는 제거할 수 없습니다.

사전 요구 사항

- **sg3_utils** 패키지를 설치합니다.

```
# dnf install sg3_utils
```

절차

1. 서버에서 기존 예약을 쿼리합니다.

```
# sg_persist --read-reservation path-to-scsi-device
```

예 7.3. /dev/sda에서 예약 쿼리

```
# *sg_persist --read-reservation /dev/sda*

  LIO-ORG   block_1          4.0
  Peripheral device type: disk
  PR generation=0x8, Reservation follows:
    Key=0x100000000000000
    scope: LU_SCOPE,  type: Exclusive Access, registrants only
```

2. *서버의 기존 등록을 제거합니다.*

```
# sg_persist --out \
        --release \
        --param-rk=reservation-key \
        --prout-type=6 \
        path-to-scsi-device
```

예 7.4. /dev/sda에서 예약 제거

```
# sg_persist --out \
        --release \
        --param-rk=0x100000000000000 \
        --prout-type=6 \
        /dev/sda

  LIO-ORG   block_1          4.0
  Peripheral device type: disk
```

추가 리소스

- ***sg_persist(8)** 매뉴얼 페이지*

8장. PNFS SCSI 레이아웃 기능 모니터링

pNFS 클라이언트와 서버가 적절한 pNFS SCSI 작업을 교환하는지 또는 일반 NFS 작업을 다시 사용할 수 있는지 모니터링할 수 있습니다.

사전 요구 사항

- pNFS SCSI 클라이언트와 서버가 구성되어 있습니다.

8.1. NFSSTAT를 사용하여 서버에서 PNFS SCSI 작업 확인

이 절차에서는 nfsstat 유틸리티를 사용하여 서버에서 pNFS SCSI 작업을 모니터링합니다.

절차

1. 서버에서 제공하는 작업을 모니터링합니다.

```
# watch --differences \
      "nfsstat --server | egrep --after-context=1 read\|write\|layout"

Every 2.0s: nfsstat --server | egrep --after-context=1 read\|write\|layout

putrootfh    read        readdir     readlink    remove  rename
2        0% 0        0% 1        0% 0        0% 0        0% 0        0%
--
setcltidconf verify   write       rellockowner bc_ctl  bind_conn
0        0% 0        0% 0        0% 0        0% 0        0% 0        0%
--
getdevlist   layoutcommit layoutget    layoutreturn secinfononam sequence
0        0% 29       1% 49       1% 5        0% 0        0% 2435     86%
```

2. 클라이언트 및 서버는 다음과 같은 경우 pNFS SCSI 작업을 사용합니다.

- layoutget,layoutreturn, layoutcommit 카운터가 증가했습니다. 즉, 서버가 레이아웃을 제공하고 있습니다.
- 서버 읽기 및 쓰기 카운터가 증가하지 않습니다. 즉, 클라이언트가 SCSI 장치에 직접 I/O 요청을 수행하고 있습니다.

8.2. MOUNTSTATS를 사용하여 클라이언트에서 PNFS SCSI 작업 확인

이 절차에서는 **/proc/self/mountstats** 파일을 사용하여 클라이언트에서 **pNFS SCSI** 작업을 모니터링합니다.

절차

1. 마운트별 작업 카운터를 나열합니다.

```
# cat /proc/self/mountstats \
    | awk /scsi_lun_0/,/^$/ \
    | egrep device\|READ\|WRITE\|LAYOUT

device 192.168.122.73:/exports/scsi_lun_0 mounted on /mnt/rhel7/scsi_lun_0 with fstype
nfs4 statvers=1.1
   nfsv4:
bm0=0xfdffbfff,bm1=0x40f9be3e,bm2=0x803,acl=0x3,sessions,pnfs=LAYOUT_SCSI
          READ: 0 0 0 0 0 0 0 0
         WRITE: 0 0 0 0 0 0 0 0
      READLINK: 0 0 0 0 0 0 0 0
       READDIR: 0 0 0 0 0 0 0 0
     LAYOUTGET: 49 49 0 11172 9604 2 19448 19454
  LAYOUTCOMMIT: 28 28 0 7776 4808 0 24719 24722
  LAYOUTRETURN: 0 0 0 0 0 0 0 0
   LAYOUTSTATS: 0 0 0 0 0 0 0 0
```

2. 결과에서 다음을 수행합니다.

- **LAYOUT** 통계는 클라이언트 및 서버가 **pNFS SCSI** 작업을 사용하는 요청을 나타냅니다.
- **READ** 및 **WRITE** 통계는 클라이언트 및 서버가 **NFS** 작업으로 대체되는 요청을 나타냅니다.

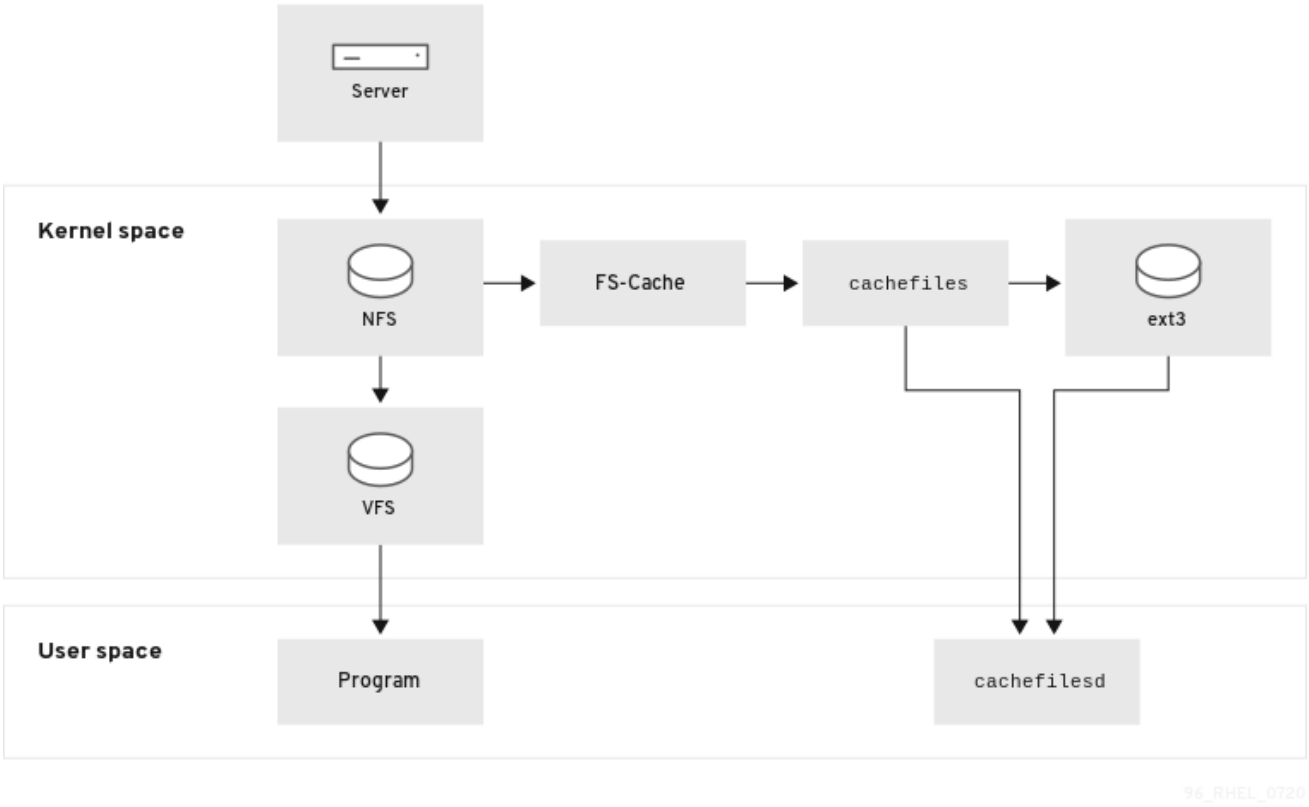
9장. FS-CACHE 시작하기

FS-Cache는 파일 시스템이 네트워크를 통해 검색된 데이터를 가져와 로컬 디스크에 캐시하는 데 사용할 수 있는 영구적인 로컬 캐시입니다. 이를 통해 네트워크를 통해 마운트된 파일 시스템의 데이터에 액세스하는 사용자의 네트워크 트래픽을 최소화할 수 있습니다(예: **NFS**).

9.1. FS-CACHE 개요

다음 다이어그램은 **FS-Cache** 작동 방법에 대한 간략한 설명입니다.

그림 **9.1. FS-Cache** 개요



FS-Cache는 시스템 관리자와 사용자에게 가능한 한 투명하도록 설계되었습니다. 솔라리스의 **cachefs**와 달리 **FS-Cache**를 사용하면 과도한 마운트된 파일 시스템을 만들지 않고도 서버의 파일 시스템이 클라이언트의 로컬 캐시와 직접 상호 작용할 수 있습니다. **NFS**를 사용하면 마운트 옵션이 클라이언트에 **FS-cache**가 활성화된 **NFS** 공유를 마운트하도록 지시합니다. 마운트 지점을 사용하면 두 커널 모듈(**fs cache** 및 **cachefiles**) 에 대한 자동 업로드가 발생합니다. **cachefilesd** 데몬은 커널 모듈과 통신하여 캐시를 구현합니다.

FS-Cache는 네트워크를 통해 작동하는 파일 시스템의 기본 작업을 변경하지 않으며, 이는 데이터를 캐시할 수 있는 영구 공간으로 파일 시스템을 제공하기만 하면 됩니다. 예를 들어 클라이언트는 **FS-Cache**가 활성화되어 있는지 여부에 관계없이 **NFS** 공유를 계속 마운트할 수 있습니다. 또한 캐시된 **NFS**는 파일

이 부분적으로 캐시될 수 있으므로 캐시에 맞지 않는 파일을 처리할 수 있으며 완전히 읽을 필요가 없습니다. **FS-Cache**는 클라이언트 파일 시스템 드라이버에서 캐시에 발생하는 모든 **I/O** 오류도 숨깁니다.

캐싱 서비스를 제공하려면 **FS-Cache**에 캐시 백엔드 가 필요합니다. 캐시 백엔드는 캐시 파일인 캐싱 서비스를 제공하도록 구성된 스토리지 드라이버입니다. 이 경우 **FS-Cache**에는 **bmap** 및 확장 속성(예: **ext3**)을 캐시 백엔드로 지원하는 마운트된 블록 기반 파일 시스템이 필요합니다.

FS-Cache 캐시 백엔드에 필요한 기능을 지원하는 파일 시스템에는 다음 파일 시스템의 **Red Hat Enterprise Linux 9** 구현이 포함됩니다.

- **ext3 (Extended attributes enabled** 포함**)**
- **ext4**
- **XFS**

FS-Cache는 네트워크를 통해 일반적으로 모든 파일 시스템을 캐시할 수 없습니다. 공유 파일 시스템의 드라이버는 **FS-Cache**, 데이터 스토리지/**retrieval**, 메타데이터 설정 및 유효성 검사와 상호 작용할 수 있도록 변경해야 합니다. **FS-Cache**는 지속성을 지원하기 위해 캐시된 파일 시스템의 인덱싱 키와 일관성 데이터를 필요로 합니다. 파일 시스템 개체와 일치하기 위한 인덱싱 키 및 일관성 데이터를 사용하여 캐시 오브젝트가 계속 유효한지 확인합니다.

참고

Red Hat Enterprise Linux 9에서 **cachefilesd** 패키지는 기본적으로 설치되지 않으므로 수동으로 설치해야 합니다.

9.2. 성능 보장

FS-Cache는 성능 향상을 보장하지 않습니다. 캐시를 사용하면 성능 저하가 발생합니다. 예를 들어 캐시된 **NFS** 공유에서는 교차 네트워크 조회에 대한 디스크 액세스 권한을 추가합니다. **FS-Cache**는 가능한 한 비동기로 시도하지만 이것이 가능하지 않은 동기 경로 (예: 읽기)가 있습니다.

예를 들어 **FS-Cache**를 사용하여 다른 차단되지 않은 **GigE** 네트워크를 통해 두 컴퓨터 간에 **NFS** 공유를 캐시하면 파일 액세스 시 성능 향상이 수행되지 않을 수 있습니다. **NFS** 요청은 로컬 디스크가 아닌 서버 메모리에서 더 빠르게 충족됩니다.

따라서 **FS-Cache**를 사용하면 다양한 요인이 손상 됩니다. 예를 들어 **FS-Cache**를 사용하여 **NFS** 트래픽을 캐시하는 경우 클라이언트의 속도가 약간 느려질 수 있지만 네트워크 대역폭을 사용하지 않고 로컬로 읽기 요청을 충족하기 때문에 네트워크 및 서버 로드가 느려질 수 있습니다.

9.3. 캐시 설정

현재 **Red Hat Enterprise Linux 9**는 **cachefiles** 캐싱 백엔드만 제공합니다. **cachefilesd** 데몬은 **cachefiles** 를 시작하고 관리합니다. **/etc/cachefilesd.conf** 파일은 **cachefiles** 에서 캐싱 서비스를 제공하는 방법을 제어합니다.

캐시 백엔드는 캐시를 호스팅하는 파티션에 특정 양의 여유 공간을 유지하여 작동합니다. 사용 가능한 공간을 사용하여 시스템의 다른 요소에 대한 응답으로 캐시를 확장 및 축소하여 루트 파일 시스템(예: 랩탑에서)을 안전하게 사용할 수 있습니다. **FS-Cache**는 캐시 큐를 통해 구성할 수 있는 이 동작의 기본값을 설정합니다. 캐시 단서 제한 구성에 대한 자세한 내용은 캐시 단서 *제한 구성* 을 참조하십시오.

다음 절차에서는 캐시를 설정하는 방법을 설명합니다.

사전 요구 사항

- **cachefilesd** 패키지가 설치되고 서비스가 성공적으로 시작됩니다. 서비스가 실행 중인지 확인하려면 다음 명령을 사용하십시오.

```
# systemctl start cachefilesd
# systemctl status cachefilesd
```

상태는 활성(실행 중) 이어야 합니다.

절차

1. 캐시 백엔드에서 캐시로 사용할 디렉터리를 구성하고 다음 매개 변수를 사용합니다.

```
$ dir /path/to/cache
```

2. 일반적으로 캐시 백엔드 디렉터리는 다음과 같이 **/etc/cachefilesd.conf** 에 **/var/cache/fscache** 로 설정됩니다.

```
$ dir /var/cache/fscache
```

3. 캐시 백엔드 디렉터리를 변경하려면 **selinux** 컨텍스트가 **/var/cache/fscache** 와 동일해야 합니다.

```
# semanage fcontext -a -e /var/cache/fscache /path/to/cache
# restorecon -Rv /path/to/cache
```

4. 캐시를 설정하는 동안 **/path/to/cache** 를 디렉토리 이름으로 바꿉니다.

5. **selinux** 컨텍스트를 설정하기 위해 지정된 명령이 작동하지 않으면 다음 명령을 사용하십시오.

```
# semanage permissive -a cachefilesd_t
# semanage permissive -a cachefiles_kernel_t
```

FS-Cache는 **/path/to/cache** 를 호스팅하는 파일 시스템에 캐시를 저장합니다. 랩탑에서는 루트 파일 시스템(/)을 호스트 파일 시스템으로 사용하는 것이 좋지만, 데스크탑 시스템의 경우 특히 캐시용으로 디스크 파티션을 마운트하는 것이 더 바람직합니다.

6. 호스트 파일 시스템은 사용자 정의 확장 속성을 지원해야 합니다. **FS-Cache**는 이러한 특성을 사용하여 일관성 유지 관리 정보를 저장합니다. **ext3** 파일 시스템(예: 장치)에 대한 사용자 정의 확장 속성을 활성화하려면 다음을 사용합니다.

```
# tune2fs -o user_xattr /dev/device
```

7. 마운트 시 파일 시스템의 확장 속성을 활성화하려면 대체 방법으로 다음 명령을 사용합니다.

```
# mount /dev/device /path/to/cache -o user_xattr
```

8. 구성 파일이 배치되면 **cachefilesd** 서비스를 시작합니다.

```
# systemctl start cachefilesd
```

9. 부팅 시 시작되도록 **cachefilesd** 를 구성하려면 **root**로 다음 명령을 실행합니다.

```
# systemctl enable cachefilesd
```

9.4. CACHE CULL 제한 구성

cachefilesd 데몬은 공유 파일 시스템의 원격 데이터를 캐싱하여 디스크 공간을 확보하여 작동합니다. 이로 인해 사용 가능한 모든 여유 공간을 소비할 수 있으므로 디스크가 루트 파티션을 배치한 경우 문제가 발생할 수 있습니다. 이를 제어하기 위해 cachefilesd 는 캐시에서 이전 오브젝트(예: 액세스 less recently)를 삭제하여 특정 양의 여유 공간을 유지하려고 합니다. 이 동작을 캐시 큐링 이라고 합니다.

캐시 큐링은 블록 백분율과 기본 파일 시스템에서 사용할 수 있는 파일의 백분율에 따라 수행됩니다. /etc/cachefilesd.conf 에는 6개의 제한을 제어하는 설정이 있습니다.

Brun N% (블록의 백분율), frun N% (파일의 백분율)

사용 가능한 공간의 양과 캐시에 있는 사용 가능한 파일 수가 이 제한보다 높으면 큐링이 해제됩니다.

bcull N% (블록 부족), fcull N% (파일의 백분율)

사용 가능한 공간 또는 캐시에 있는 파일 수가 이러한 제한 중 하나 미만이면 큐링이 시작됩니다.

bstop N% (블록당량), fstop N% (파일의 백분율)

사용 가능한 공간 또는 캐시에 있는 파일 수가 이러한 제한 중 하나 미만으로 떨어지면 큐로 인해 이러한 제한을 초과할 때까지 디스크 공간이나 파일을 더 이상 할당할 수 없습니다.

각 설정 의 기본값은 다음과 같습니다.

- **Brun/frun - 10%**
- **bcull/fcull - 7%**
- **bstop/fstop - 3%**

이러한 설정을 구성할 때 다음 사항이 충족되어야 합니다.

- **0 bstop < bcull < brun < 100**
- **0 fstop < fcull < frun < 100**

다음은 사용 가능한 공간 및 사용 가능한 파일의 백분율이며 **df** 프로그램에서 표시하는 백분율을 **100**개로 표시하지 않습니다.

중요

큐링은 **bxxx** 및 **fxxx** 쌍 모두에 동시에 의존합니다. 사용자는 이를 별도로 처리할 수 없습니다.

9.5. FSCACHE 커널 모듈에서 통계 정보 검색

FS-Cache는 일반 통계 정보를 추적합니다. 다음 절차에서는 이 정보를 얻는 방법을 설명합니다.

절차

1. **FS-Cache**에 대한 통계 정보를 보려면 다음 명령을 사용하십시오.

```
# cat /proc/fs/fscache/stats
```

FS-Cache 통계에는 의사 결정 지점 및 개체 카운터에 대한 정보가 포함됩니다. 자세한 내용은 다음 커널 문서를 참조하십시오.

/usr/share/doc/kernel-doc-4.18.0/Documentation/filesystems/caching/fscache.txt

9.6. FS-CACHE 참조

이 섹션에서는 **FS-Cache**에 대한 참조 정보를 제공합니다.

1. **cachefilesd** 및 구성 방법에 대한 자세한 내용은 **man cachefilesd** 및 **man cachefilesd.conf**를 참조하십시오. 다음 커널 문서에서도 추가 정보를 제공합니다.

   - **/usr/share/doc/cachefilesd/README**
   - **/usr/share/man/man5/cachefilesd.conf.5.gz**

- ***/usr/share/man/man8/cachefilesd.8.gz***

2. *디자인 제약 조건, 사용 가능한 통계 및 기능에 대한 세부 정보를 포함하여* ***FS-Cache****에 대한 일반적인 내용은 다음 커널 문서를 참조하십시오.*

   /usr/share/doc/kernel-doc-4.18.0/Documentation/filesystems/caching/fscache.txt

10장. NFS에서 캐시 사용

명시적으로 지시하지 않는 한 NFS는 캐시를 사용하지 않습니다. 이 단락에서는 FS-Cache를 사용하여 NFS 마운트를 구성하는 방법을 보여줍니다.

사전 요구 사항

- cachefilesd 패키지가 설치되어 실행 중입니다. 실행 중인지 확인하려면 다음 명령을 사용하십시오.

```
# systemctl start cachefilesd
# systemctl status cachefilesd
```

  상태는 활성(실행 중) 이어야 합니다.

- 다음 옵션을 사용하여 NFS 공유를 마운트합니다.

```
# mount nfs-share:/ /mount/point -o fsc
```

  직접 I/O 또는 쓰기를 위해 파일을 열지 않는 한 /mount/point 아래의 파일에 대한 모든 액세스는 캐시를 통과합니다. 자세한 내용은 NFS의 캐시 제한 사항을 참조하십시오.

NFS 색인은 파일 이름이 아닌 NFS 파일 핸들을 사용하여 콘텐츠를 캐시합니다. 따라서 하드 연결된 파일이 캐시를 올바르게 공유합니다.

NFS 버전 3, 4.0, 4.1 및 4.2는 캐싱을 지원합니다. 그러나 각 버전은 캐싱에 서로 다른 분기를 사용합니다.

10.1. NFS 캐시 공유 구성

NFS 캐시 공유에는 몇 가지 잠재적인 문제가 있습니다. 캐시는 영구적이므로 캐시의 데이터 블록은 다음 네 가지 키 순서로 인덱싱됩니다.

- 수준 1: 서버 세부 정보

- *수준 **2** 일부 마운트 옵션, 보안 유형, **FSID; uniquifier***
- *수준 **3:** 파일 처리*
- *수준 **4:** 파일의 페이지 번호*

*수퍼 블록 간의 일관성 관리 문제를 방지하기 위해 데이터를 캐시하는 데 필요한 모든 **NFS** 수퍼 블록에는 고유한 레벨 **2** 키가 있습니다. 일반적으로 동일한 소스 볼륨 및 옵션을 사용하는 두 **NFS** 마운트는 수퍼 블록을 공유하므로 해당 볼륨 내에서 다른 디렉터리를 마운트하더라도 캐싱을 공유합니다.*

다음은 다양한 옵션을 사용하여 캐시 공유를 구성하는 예제입니다.

절차

1. *다음 명령을 사용하여 **NFS** 공유를 마운트합니다.*

```
mount home0:/disk0/fred /home/fred -o fsc
mount home0:/disk0/jim /home/jim -o fsc
```

*여기에서 **/home/fred** 및 **/home/jim** 은 특히 **NFS** 서버(**home0**)에서 동일한 볼륨/파티션에서 발생하는 경우 수퍼 블록을 공유할 수 있습니다.*

2. *수퍼 블록을 공유하지 않으려면 **mount** 명령을 다음 옵션과 함께 사용하십시오.*

```
mount home0:/disk0/fred /home/fred -o fsc,rsize=8192
mount home0:/disk0/jim /home/jim -o fsc,rsize=65536
```

*이 경우 **/home/fred** 및 **/home/jim** 은 수준 **2** 키의 일부인 다른 네트워크 액세스 매개 변수를 가지고 있으므로 수퍼 블록을 공유하지 않습니다.*

3. *수퍼 블록을 공유하지 않고 두 개의 하위 트리(**/home/fred1** 및 **/home/fred2**)의 콘텐츠를 두 번 캐시하려면 다음 명령을 사용하십시오.*

```
mount home0:/disk0/fred /home/fred1 -o fsc,rsize=8192
mount home0:/disk0/fred /home/fred2 -o fsc,rsize=65536
```

4.

수퍼 블록 공유를 방지하는 또 다른 방법은 **nosharecache** 매개 변수를 사용하여 명시적으로 억제하는 것입니다. 동일한 예제를 사용합니다.

```
mount home0:/disk0/fred /home/fred -o nosharecache,fsc
mount home0:/disk0/jim /home/jim -o nosharecache,fsc
```

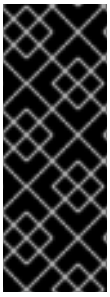
그러나 이 경우 **home0:/disk0/fred** 및 **home0:/disk0/jim** 의 수준 **2** 키를 구분할 수 없으므로 수퍼 블록 중 하나만 캐시를 사용할 수 있습니다.

5.

수퍼 블록에 주소 지정을 지정하려면 마운트 중 하나 이상(예: **fsc=unique-identifier**)에 고유 식별자 를 추가합니다.

```
mount home0:/disk0/fred /home/fred -o nosharecache,fsc
mount home0:/disk0/jim /home/jim -o nosharecache,fsc=jim
```

여기서는 고유 식별자 **jim** 이 **/home/jim** 의 캐시에 사용된 레벨 **2** 키에 추가됩니다.



중요

사용자는 다른 통신 또는 프로토콜 매개 변수가 있는 수퍼 블록 간에 캐시를 공유할 수 없습니다. 예를 들어 **NFSv4.0**과 **NFSv3** 간 또는 **NFSv4.1**과 **NFSv4.2** 간에 다른 수퍼 블록을 강제로 공유하므로 공유할 수 없습니다. 읽기 크기(**rsize**)와 같은 매개 변수를 설정하면 다른 수퍼 블록을 강제 적용하므로 캐시 공유를 방지할 수 있습니다.

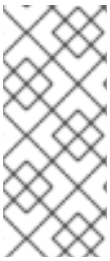
10.2. NFS를 사용한 캐시 제한 사항

NFS에는 몇 가지 캐시 제한 사항이 있습니다.

- 직접 **I/O**에 대해 공유 파일 시스템에서 파일을 열면 캐시를 자동으로 무시합니다. 이러한 유형의 액세스는 서버로 직접 전송되어야 하기 때문입니다.
- 직접 **I/O**를 위해 공유 파일 시스템에서 파일을 열거나 파일 캐시된 복사본을 플러시합니다. **FS-Cache**는 직접 **I/O** 또는 쓰기를 위해 더 이상 열 수 없을 때까지 파일을 다시 캐시하지 않습니다.
- 또한 **FS-Cache**의 이 릴리스는 일반 **NFS** 파일만 캐시합니다. **FS-Cache**는 디렉토리, 심볼릭 링크, 장치 파일, **firstFOs** 및 소켓을 캐시 하지 않습니다.

11장. RED HAT ENTERPRISE LINUX에서 SMB 공유 마운트

SMB(Server Message Block) 프로토콜은 파일 공유 및 공유 프린터와 같은 서버의 리소스에 액세스하는 데 사용되는 애플리케이션 계층 네트워크 프로토콜을 구현합니다.



참고

SMB의 컨텍스트에서는 SMB의 대화 상자인 CIFS(Common Internet File System) 프로토콜에 대해 언급할 수 있습니다. SMB 및 CIFS 프로토콜이 모두 지원되며, SMB 및 CIFS 공유 마운트에 포함된 커널 모듈과 유틸리티에는 모두 cifs 라는 이름이 사용됩니다.

이 섹션에서는 SMB 서버에서 공유를 마운트하는 방법에 대해 설명합니다.

사전 요구 사항

Microsoft Windows에서는 SMB가 기본적으로 구현됩니다. Red Hat Enterprise Linux에서 커널의 cifs.ko 파일 시스템 모듈은 SMB 공유 마운트를 지원합니다. 따라서 cifs-utils 패키지를 설치합니다.

```
# dnf install cifs-utils
```

cifs-utils 패키지는 다음을 수행할 수 있는 유틸리티를 제공합니다.

- SMB 및 CIFS 공유 마운트
- 커널의 인증 키에서 NT Lan Manager(NTLM) 자격 증명을 관리합니다.
- SMB 및 CIFS 공유의 보안 설명자에서 ACL(액세스 제어 목록) 설정 및 표시

11.1. 지원되는 SMB 프로토콜 버전

cifs.ko 커널 모듈은 다음과 같은 SMB 프로토콜 버전을 지원합니다.

- SMB 1

주의

SMB1 프로토콜은 알려진 보안 문제로 인해 더 이상 사용되지 않으며 사설 네트워크에서만 사용할 수 있습니다. **SMB1**이 여전히 지원되는 옵션으로 제공되는 주요 이유는 현재 **UNIX** 확장을 지원하는 유일한 **SMB** 프로토콜 버전이라는 것입니다. **SMB**에서 **UNIX** 확장을 사용할 필요가 없는 경우 **SMB2** 이상을 사용하는 것이 좋습니다.

- **SMB 2.0**
- **SMB 2.1**
- **SMB 3.0**
- **SMB 3.1.1**

참고

프로토콜 버전에 따라 일부 **SMB** 기능이 구현되는 것은 아닙니다.

11.2. UNIX 확장 지원

Samba는 **SMB** 프로토콜에서 **CAP_UNIX** 기능 비트를 사용하여 **UNIX** 확장 기능을 제공합니다. 이러한 확장 기능은 **cifs.ko** 커널 모듈에서도 지원됩니다. 그러나 **Samba**와 커널 모듈 모두 **SMB 1** 프로토콜에서만 **UNIX** 확장을 지원합니다.

UNIX 확장을 사용하려면 다음을 수행합니다.

1. **/etc/hiera/hiera.conf** 파일의 **[global]** 섹션에서 **server min protocol** 매개 변수를 **NT1** 로 설정합니다.
2. **mount** 명령에 **-o vers=1.0** 옵션을 제공하여 **SMB 1** 프로토콜을 사용하여 공유를 마운트합니

다. 예를 들면 다음과 같습니다.

```
# mount -t cifs -o vers=1.0,username=user_name //server_name/share_name /mnt/
```

기본적으로 커널 모듈은 **SMB 2** 또는 서버에서 지원하는 가장 높은 이후의 프로토콜 버전을 사용합니다. **-o vers=1.0** 옵션을 **mount** 명령에 전달하면 커널 모듈에서 **UNIX** 확장을 사용하는 데 필요한 **SMB 1** 프로토콜을 사용하도록 강제 적용합니다.

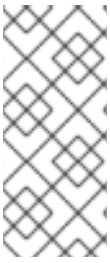
UNIX 확장 기능이 활성화되어 있는지 확인하려면 마운트된 공유 옵션을 표시합니다.

```
# mount
...
//server/share on /mnt type cifs (...,unix,...)
```

마운트 옵션 목록에 **unix** 항목이 표시되면 **UNIX** 확장이 활성화됩니다.

11.3. SMB 공유 수동 마운트

SMB 공유만 임시로 마운트해야 하는 경우 마운트 유틸리티를 사용하여 수동으로 마운트할 수 있습니다.



참고

시스템을 재부팅할 때 수동으로 마운트된 공유는 자동으로 다시 마운트되지 않습니다. **Red Hat Enterprise Linux**가 시스템 부팅 시 공유를 자동으로 마운트하도록 구성하려면 시스템 부팅 시 **SMB** 공유 마운트를 자동으로 참조하십시오.

사전 요구 사항

- **cifs-utils** 패키지가 설치되어 있습니다.

절차

SMB 공유를 수동으로 마운트하려면 **-t cifs** 매개 변수와 함께 **mount** 유틸리티를 사용합니다.

```
# mount -t cifs -o username=user_name //server_name/share_name /mnt/
Password for user_name@//server_name/share_name:  password
```

-o 매개변수에서는 공유를 마운트하는 데 사용되는 옵션을 지정할 수 있습니다. 자세한 내용은 mount.cifs(8) 도움말 페이지 및 자주 사용되는 마운트 옵션 의 OPTIONS 섹션을 참조하십시오.

> *예 11.1. 암호화된 SMB 3.0 연결을 사용하여 공유 마운트*
>
> *암호화된 SMB 3.0 연결을 통해 /mnt/ 디렉터리에 대해 \\server\example\ 공유를 DOMAIN\Administrator 사용자로 마운트하려면 다음을 실행합니다.*
>
> ```
> # mount -t cifs -o username=DOMAIN\Administrator,seal,vers=3.0 //server/example /mnt/
> Password for DOMAIN\Administrator@//server_name/share_name:  password
> ```

11.4. 시스템이 부팅될 때 SMB 공유를 자동으로 마운트

서버에 마운트된 SMB 공유에 대한 액세스가 영구적으로 필요한 경우 부팅 시 공유를 자동으로 마운트합니다.

사전 요구 사항

- *cifs-utils 패키지가 설치되어 있습니다.*

절차

시스템이 부팅될 때 SMB 공유를 자동으로 마운트하려면 /etc/fstab 파일에 공유에 대한 항목을 추가합니다. 예를 들면 다음과 같습니다.

```
//server_name/share_name  /mnt  cifs  credentials=/root/smb.cred  0 0
```

중요

시스템이 공유를 자동으로 마운트할 수 있도록 하려면 사용자 이름, 암호 및 도메인 이름을 자격 증명 파일에 저장해야 합니다. 자세한 내용은 자격 증명 파일을 사용하여 SMB 공유 인증을 참조하십시오.

/etc/fstab 에 있는 행의 네 번째 필드에서 자격 증명 파일 경로와 같은 마운트 옵션을 지정합니다. 자세한 내용은 mount.cifs(8) 도움말 페이지 및 자주 사용되는 마운트 옵션 의 OPTIONS 섹션을 참조하십시오.

공유가 성공적으로 마운트되는지 확인하려면 다음을 입력합니다.

```
# mount /mnt/
```

11.5. 자격 증명 파일을 사용하여 SMB 공유에 인증

부팅 시 공유를 자동으로 마운트하는 경우와 같은 특정 상황에서는 사용자 이름 및 암호를 입력하지 않고 공유를 마운트해야 합니다. 이를 구현하려면 자격 증명 파일을 만듭니다.

사전 요구 사항

- cifs-utils 패키지가 설치되어 있습니다.

절차

1. /root/subject.cred 과 같은 파일을 생성하고 해당 파일의 사용자 이름, 암호 및 도메인 이름을 지정합니다.

```
username=user_name
password=password
domain=domain_name
```

2. 소유자만 파일에 액세스할 수 있도록 권한을 설정합니다.

```
# chown user_name /root/smb.cred
# chmod 600 /root/smb.cred
```

이제 credentials=file_name 마운트 옵션을 마운트 유틸리티에 전달하거나 /etc/fstab 파일에서 사용하여 사용자 이름 및 암호를 입력하라는 메시지가 표시되지 않고 공유를 마운트할 수 있습니다.

11.6. 자주 사용되는 마운트 옵션

SMB 공유를 마운트할 때 마운트 옵션이 결정됩니다.

- 서버와의 연결이 설정되는 방법입니다. 예를 들어 서버에 연결할 때 사용되는 SMB 프로토콜 버전은 무엇입니까.

- *공유가 로컬 파일 시스템에 마운트되는 방법. 예를 들어 시스템이 원격 파일 및 디렉터리 권한을 재정의하여 여러 로컬 사용자가 서버의 콘텐츠에 액세스할 수 있도록 합니다.*

/etc/fstab 파일의 네 번째 필드 또는 mount 명령의 -o 매개 변수에 여러 옵션을 설정하려면 쉼표로 구분합니다. 예를 들어, multiuser 옵션을 사용하여 공유 마운트를 참조하십시오.

다음 목록은 자주 사용되는 마운트 옵션을 제공합니다.

옵션	설명
credentials=*file_name*	자격 증명 파일의 경로를 설정합니다. 자격 증명 파일을 사용하여 SMB 공유 인증을 참조하십시오.
dir_mode=*mode*	서버가 CIFS UNIX 확장을 지원하지 않는 경우 디렉터리 모드를 설정합니다.
file_mode=*mode*	서버가 CIFS UNIX 확장을 지원하지 않는 경우 파일 모드를 설정합니다.
password=*password*	SMB 서버에 인증하는 데 사용되는 암호를 설정합니다. 또는 **credentials** 옵션을 사용하여 자격 증명 파일을 지정합니다.
seal	SMB 3.0 또는 이후 프로토콜 버전을 사용하여 연결에 대한 암호화 지원을 활성화합니다. 따라서 **vers** 마운트 옵션과 함께 **seal** 을 **3.0** 이상으로 설정합니다. SMB 공유 수동 마운트의 예를 참조하십시오.
sec=*security_mode*	ntlmsspi . **ntlmsspi** 와 같은 보안 모드를 설정하여 RuntimeClassv2 암호 해시 및 활성화된 패킷 서명을 활성화합니다. 지원되는 값 목록은 **mount.cifs(8)** 도움말 페이지의 옵션 설명을 참조하십시오. 서버가 **ntlmv2** 보안 모드를 지원하지 않는 경우 기본값인**title=ntlmssp** 를 사용합니다. 보안상의 이유로 비보안 **ntlm** 보안 모드를 사용하지 마십시오.
username=*user_name*	SMB 서버에 인증하는 데 사용되는 사용자 이름을 설정합니다. 또는 **credentials** 옵션을 사용하여 자격 증명 파일을 지정합니다.
vers=*SMB_protocol_version*	서버와의 통신에 사용되는 SMB 프로토콜 버전을 설정합니다.

전체 목록은 mount.cifs(8) 도움말 페이지의 OPTIONS 섹션을 참조하십시오.

12장. 다중 사용자 SMB 마운트 수행

공유를 마운트하기 위해 제공하는 인증 정보에는 기본적으로 마운트 지점에 대한 액세스 권한이 결정됩니다. 예를 들어 공유를 마운트할 때 **DOMAIN\example** 사용자를 사용하는 경우 어떤 로컬 사용자가 작업을 수행하는지 관계없이 공유의 모든 작업이 이 사용자로 실행됩니다.

그러나 특정 상황에서는 관리자가 시스템을 부팅할 때 공유를 자동으로 마운트하려고 하지만 사용자는 자신의 자격 증명을 사용하여 공유 콘텐츠에 대해 작업을 수행해야 합니다. 다중 사용자 마운트 옵션을 사용하면 이 시나리오를 구성할 수 있습니다.

중요

다중 사용자 마운트 옵션을 사용하려면 **Citadel 5** 또는 자격 증명 파일이 있는 **ntlmssp** 옵션과 같이 비대화형 방식으로 자격 증명 제공을 지원하는 보안 유형으로도 **rootfs** 마운트 옵션을 추가로 설정해야 합니다. 자세한 내용은 사용자로 공유 액세스를 참조하십시오.

루트 사용자는 다중 사용자 옵션을 사용하여 공유를 마운트하고 공유 콘텐츠에 대한 최소 액세스 권한이 있는 계정을 생성합니다. 그런 다음 일반 사용자는 **cifscreds** 유틸리티를 사용하여 사용자 이름과 암호를 현재 세션의 커널 인증 키에 제공할 수 있습니다. 사용자가 마운트된 공유의 콘텐츠에 액세스하는 경우 커널은 공유를 마운트하는 데 처음 사용된 커널 인증 키 대신 커널 인증 키의 자격 증명을 사용합니다.

이 기능을 사용하면 다음 단계로 구성됩니다.

- 다중 사용자 옵션을 사용하여 공유를 마운트합니다 .
- 필요한 경우 다중 사용자 옵션을 사용하여 공유가 성공적으로 마운트되었는지 확인합니다.
- 공유에 사용자로 액세스합니다.

사전 요구 사항

- **cifs-utils** 패키지가 설치되어 있습니다.

12.1. 다중 사용자 옵션을 사용하여 공유 마운트

사용자가 자신의 자격 증명을 사용하여 공유에 액세스하기 전에 제한된 권한이 있는 계정을 사용하여 **root** 사용자로 공유를 마운트합니다.

절차

시스템이 부팅될 때 다중 사용자 옵션을 사용하여 공유를 자동으로 마운트하려면 다음을 수행합니다.

1. **/etc/fstab** 파일에 공유 항목을 만듭니다. 예를 들면 다음과 같습니다.

```
//server_name/share_name  /mnt  cifs
multiuser,sec=ntlmssp,credentials=/root/smb.cred  0 0
```

2. 공유를 마운트합니다.

```
# mount /mnt/
```

시스템이 부팅될 때 공유를 자동으로 마운트하지 않으려면 **-oResourceOverride,sec=security_type** 을 **mount** 명령에 전달하여 수동으로 마운트합니다. 수동으로 **SMB** 공유 마운트에 대한 자세한 내용은 **SMB** 공유 수동으로 마운트를 참조하십시오.

12.2. 다중 사용자 옵션을 사용하여 SMB 공유가 마운트되었는지 확인

다중 사용자 옵션을 사용하여 공유가 마운트되었는지 확인하려면 마운트 옵션을 표시합니다.

절차

```
# mount
...
//server_name/share_name on /mnt type cifs (sec=ntlmssp,multiuser,...)
```

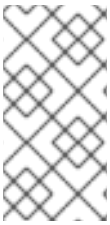
마운트 옵션 목록에 다중 사용자 항목이 표시되면 기능이 활성화됩니다.

12.3. 사용자로 공유에 액세스

*사용자 지정 옵션을 사용하여 **SMB** 공유를 마운트하면 사용자는 해당 서버에 대한 자격 증명을 커널의 인증 키에 제공할 수 있습니다.*

```
# cifscreds add -u SMB_user_name server_name
Password: password
```

*사용자가 마운트된 **SMB** 공유가 포함된 디렉터리에서 작업을 수행할 때, 서버는 공유가 마운트될 때 처음 사용된 대신 이 사용자에 대한 파일 시스템 권한을 적용합니다.*



참고

여러 사용자가 마운트된 공유에서 자체 자격 증명을 사용하여 동시에 작업을 수행할 수 있습니다.

13장. 영구 이름 지정 속성 개요

시스템 관리자는 여러 시스템 부팅에 대해 안정적인 스토리지 설정을 빌드하기 위해 영구 이름 지정 속성을 사용하여 스토리지 볼륨을 참조해야 합니다.

13.1. 비영구 이름 지정 특성의 단점

Red Hat Enterprise Linux는 스토리지 장치를 식별하는 다양한 방법을 제공합니다. 특히 드라이브를 설치하거나 다시 포맷할 때 실수로 잘못된 장치에 액세스하지 않도록 각 장치를 사용하기 위해 올바른 옵션을 사용하는 것이 중요합니다.

일반적으로 **/dev/sd(major number)** 의 형태로 비영구적인 이름은 **Linux**에서 스토리지 장치를 참조하는 데 사용됩니다. 주요 및 마이너 번호 범위 및 관련 **sd** 이름은 감지 시 각 장치에 할당됩니다. 즉, 장치 탐지 순서가 변경될 경우 메이저 및 마이너 번호 범위와 연결된 **sd** 이름 간의 연결이 변경될 수 있습니다.

이러한 순서의 변경은 다음과 같은 상황에서 발생할 수 있습니다.

- 시스템 부팅 프로세스의 병렬화는 시스템 부팅마다 다른 순서로 스토리지 장치를 탐지합니다.
- 디스크가 **SCSI** 컨트롤러의 전원을 켜거나 응답하지 않습니다. 그러면 일반 장치 프로브에 의해 검색되지 않습니다. 시스템에서 디스크에 액세스할 수 없으며 후속 장치는 연결된 **sd** 이름이 아래로 전환됨을 포함하여 주요 및 마이너 번호 범위를 갖습니다. 예를 들어 일반적으로 **sdb** 라고 하는 디스크가 감지되지 않으면 일반적으로 **sdc** 라고 하는 디스크가 **sdb** 로 표시됩니다.
- **SCSI** 컨트롤러(호스트 버스 어댑터 또는 **HBA**)가 초기화되지 않아 해당 **HBA**에 연결된 모든 디스크가 탐지되지 않습니다. 이후 프로브된 **HBA**에 연결된 모든 디스크에는 다른 메이저 및 마이너 번호 범위 및 관련 **sd** 이름이 할당됩니다.
- 시스템에 다른 유형의 **HBA**가 있는 경우 드라이버 초기화 순서가 변경됩니다. 이로 인해 해당 **HBA**에 연결된 디스크가 다른 순서로 감지됩니다. 이는 **HBA**가 시스템의 다른 **PCI** 슬롯으로 이동되는 경우에도 발생할 수 있습니다.
- 파이버 채널, **iSCSI** 또는 **FCoE** 어댑터로 시스템에 연결된 디스크는 스토리지 어레이로 또는 전원이 꺼지는 스위치로 인해 스토리지 장치를 프로브할 때 액세스할 수 없습니다. 이는 스토리지 어레이가 시스템을 부팅하는 데 걸리는 시간보다 시간이 오래 걸리는 경우 정전 후 시스템을 재부팅할 때 발생할 수 있습니다. 일부 파이버 채널 드라이버는 **WWPN** 매핑에 영구 **SCSI** 대상 **ID**를

지정하는 메커니즘을 지원하지만, 이로 인해 메이저 및 마이너 번호 범위가 없고 관련 **sd** 이름은 예약되지 않습니다. 이는 일관된 **SCSI** 대상 **ID** 번호만 제공합니다.

이러한 이유로 **/etc/fstab** 파일에서와 같이 장치를 참조할 때 메이저 및 마이너 번호 범위 또는 관련 **sd** 이름을 사용하지 않는 것이 좋습니다. 잘못된 장치가 마운트될 가능성이 있으며 데이터 손상이 발생할 수 있습니다.

그러나 장치에서 오류를 보고할 때와 같이 다른 메커니즘을 사용하는 경우에도 **sd** 이름을 참조해야 하는 경우가 있습니다. 이는 **Linux** 커널이 장치와 관련된 커널 메시지에서 **sd** 이름(및 **SCSI host/channel/target/LUN tuples**)을 사용하기 때문입니다.

13.2. 파일 시스템 및 장치 식별자

이 섹션에서는 파일 시스템과 블록 장치를 식별하는 영구 속성의 차이점을 설명합니다.

파일 시스템 식별자

파일 시스템 식별자는 블록 장치에서 생성된 특정 파일 시스템에 연결됩니다. 식별자는 파일 시스템의 일부로 저장됩니다. 파일 시스템을 다른 장치로 복사하는 경우에도 동일한 파일 시스템 식별자를 전달합니다. 반면, **mkfs** 유틸리티로 포맷하는 등의 장치를 다시 작성하는 경우 장치는 속성이 손실됩니다.

파일 시스템 식별자는 다음과 같습니다.

- 고유 식별자(**UUID**)
- 레이블

장치 식별자

장치 식별자는 블록 장치에 연결됩니다(예: 디스크 또는 파티션). **mkfs** 유틸리티로 포맷하는 등의 장치를 다시 작성하는 경우 장치는 파일 시스템에 저장되지 않기 때문에 속성을 유지합니다.

장치 식별자는 다음과 같습니다.

- **World Wide Identifier (WWID)**

- 파티션 UUID
- 일련 번호

권장 사항

- 논리 볼륨과 같은 일부 파일 시스템은 여러 장치에 걸쳐 있습니다. Red Hat은 장치 식별자가 아닌 파일 시스템 식별자를 사용하여 이러한 파일 시스템에 액세스하는 것을 권장합니다.

13.3. /DEV/DISK/의 UDEV 메커니즘에 의해 관리되는 장치 이름

이 섹션에는 udev 서비스가 /dev/disk/ 디렉터리에 제공하는 다양한 종류의 영구 이름 지정 속성이 나열되어 있습니다.

udev 메커니즘은 스토리지 장치뿐만 아니라 Linux의 모든 유형의 장치에 사용됩니다. 스토리지 장치의 경우 Red Hat Enterprise Linux에는 /dev/disk/ 디렉터리에 심볼릭 링크를 생성하는 udev 규칙이 포함되어 있습니다. 이를 통해 스토리지 장치를 참조할 수 있습니다.

- 해당 콘텐츠
- 고유 식별자
- 일련 번호

udev 이름 지정 속성은 영구적이지만 시스템 재부팅 시 자체적으로 변경되지 않는 경우에도 일부 속성을 구성할 수 있습니다.

13.3.1. 파일 시스템 식별자

/dev/disk/by-uuid/의 UUID 속성

이 디렉터리의 항목은 장치에 저장된 콘텐츠(즉, 데이터)의 고유 식별자 (UUID)에 의해 스토리지 장치를 참조하는 심볼릭 이름을 제공합니다. 예를 들면 다음과 같습니다.

```
/dev/disk/by-uuid/3e6be9de-8139-11d1-9106-a43f08d823a6
```

UUID를 사용하여 다음 구문으로 /etc/fstab 파일의 장치를 참조할 수 있습니다.

```
UUID=3e6be9de-8139-11d1-9106-a43f08d823a6
```

파일 시스템을 생성할 때 UUID 특성을 구성할 수 있으며 나중에 변경할 수도 있습니다.

/dev/disk/by-label/의 Label 속성

이 디렉터리의 항목은 장치에 저장된 콘텐츠(즉, 데이터)의 레이블 로 스토리지 장치를 참조하는 심볼릭 이름을 제공합니다.

예를 들면 다음과 같습니다.

```
/dev/disk/by-label/Boot
```

레이블을 사용하여 다음 구문으로 /etc/fstab 파일의 장치를 참조할 수 있습니다.

```
LABEL=Boot
```

파일 시스템을 생성할 때 Label 속성을 구성할 수 있으며 나중에 변경할 수도 있습니다.

13.3.2. 장치 식별자

/dev/disk/by-id/의 WWID 속성

WWID(Global Wide Identifier)는 SCSI 표준에 필요한 영구적인 시스템 독립적인 식별자 입니다. WWID 식별자는 모든 스토리지 장치에 대해 고유하도록 보장되며 장치에 액세스하는 데 사용되는 경로와는 독립적입니다. 식별자는 장치의 속성이지만 장치의 콘텐츠(즉, 데이터)에 저장되지 않습니다.

이 식별자는 장치 식별 바이탈 제품 데이터(페이지 0x83) 또는 단위 일련 번호(페이지 0x80)를 검색하기 위해 SCSI 후에 얻을 수 있습니다.

Red Hat Enterprise Linux는 WWID 기반 장치 이름과 적절한 매핑을 해당 시스템의 현재 /dev/sd 이름으로 자동 유지 관리합니다. 애플리케이션은 다른 시스템에서 장치에 액세스하는 경우에도 /dev/disk/by-id/ 이름을 사용하여 디스크의 데이터를 참조할 수 있습니다.

예 13.1. WWID 매핑

WWID 심볼릭 링크	비영구 장치	참고
/dev/disk/by-id/scsi-3600508b400105e210000900000490000	**/dev/sda**	페이지 **0x83** 식별자가 있는 장치
/dev/disk/by-id/scsi-SSEAGATE_ST373453LW_3HW1RHM6	**/dev/sdb**	**0x80** 페이지의 식별자가 있는 장치
/dev/disk/by-id/ata-SAMSUNG_MZNLN256HMHQ-000L7_S2WDNX0J336519-part3	**/dev/sdc3**	디스크 파티션

*시스템에서 제공하는 이러한 영구 이름 외에도 **udev** 규칙을 사용하여 스토리지의 **WWID**에 매핑된 고유한 이름을 구현할 수도 있습니다.*

***/dev/disk/by-partuuid*의 *Partition UUID* 속성**

***Partition UUID(PARTUUID)** 속성은 **GPT** 파티션 테이블에 정의된 대로 파티션을 식별합니다.*

*예 **13.2.** 파티션 **UUID** 매핑*

PARTUUID symlink	비영구 장치
/dev/disk/by-partuuid/4cd1448a-01	**/dev/sda1**
/dev/disk/by-partuuid/4cd1448a-02	**/dev/sda2**
/dev/disk/by-partuuid/4cd1448a-03	**/dev/sda3**

***/dev/disk/by-path/*의 *Path* 속성**

이 속성은 장치에 액세스하는 데 사용되는 하드웨어 경로에서 스토리지 장치를 참조하는 심볼릭 이름을 제공합니다.

*하드웨어 경로의 일부(예: **PCI ID**, 대상 포트 또는 **LUN** 번호)가 변경되면 **Path** 속성이 실패합니다. 따라서 **Path** 속성은 신뢰할 수 없습니다. 그러나 **Path** 속성은 다음 시나리오 중 하나에서 유용할 수 있습니다.**However, the Path attribute may be useful in one of the following scenarios:***

- *나중에 교체할 디스크를 식별해야 합니다.*

- 특정 위치의 디스크에 스토리지 서비스를 설치하려고 합니다.

13.4. DM MULTIPATH를 사용하는 WORLD WIDE IDENTIFIER

WWID(WWID)와 비영구 장치 이름 간에 매핑되도록 DM(Device Mapper) Multipath를 구성할 수 있습니다.

시스템에서 장치로 경로가 여러 개 있는 경우 DM Multipath는 WWID를 사용하여 이를 감지합니다. 그런 다음 DM Multipath는 /dev/mapper/wwid 디렉터리에 /dev/mapper/3600508b400105df70000e00000ac0000 에 하나의 "pseudo-device"를 표시합니다.

multipath -l 명령은 비영구 식별자에 대한 매핑을 표시합니다.

- Host:Channel:Target:LUN
- /dev/sd name
- 주:마이너 번호

예 13.3. 다중 경로 구성의 WWID 매핑

multipath -l 명령의 출력 예:

```
3600508b400105df70000e00000ac0000 dm-2 vendor,product
[size=20G][features=1 queue_if_no_path][hwhandler=0][rw]
\_ round-robin 0 [prio=0][active]
 \_ 5:0:1:1 sdc 8:32  [active][undef]
 \_ 6:0:1:1 sdg 8:96  [active][undef]
\_ round-robin 0 [prio=0][enabled]
 \_ 5:0:0:1 sdb 8:16  [active][undef]
 \_ 6:0:0:1 sdf 8:80  [active][undef]
```

DM Multipath는 각 WWID 기반 장치 이름을 시스템의 해당 /dev/sd 이름에 자동으로 매핑합니다. 이러한 이름은 경로 변경 전반에 걸쳐 지속되며 다른 시스템의 장치에 액세스할 때 일관되게 유지됩니다.

DM Multipath의 user_friendly_names 기능을 사용하면 WWID가 /dev/mapper/mpathN 형식의 이름에 매핑됩니다. 기본적으로 이 매핑은 /etc/multipath/bindings 파일에서 유지됩니다. 이러한 mpathN 이름은 파일이 유지되는 한 지속됩니다.

중요

user_friendly_names 를 사용하는 경우 클러스터에서 일관된 이름을 얻으려면 추가 단계가 필요합니다.

13.5. UDEV 장치 이름 지정 규칙의 제한 사항

다음은 udev 이름 지정 규칙의 몇 가지 제한 사항입니다.

- udev 메커니즘이 udev 이벤트에 대해 처리될 때 스토리지 장치를 쿼리하는 기능에 의존할 수 있으므로 쿼리가 수행되는 시점에서 장치에 액세스할 수 없습니다. 이는 장치가 서버 새시에 없는 경우 파이버 채널, iSCSI 또는 FCoE 스토리지 장치에서 더 발생할 가능성이 높습니다.
- 커널은 udev 이벤트를 언제든지 전송하여 규칙을 처리하여 장치에 액세스할 수 없는 경우 /dev/disk/by-*/ 링크가 제거될 수 있습니다.
- udev 이벤트가 생성될 때와 같이 많은 장치가 탐지되고 사용자 공간 udevd 서비스에서 각 이벤트에 대한 규칙을 처리하는 데 약간의 시간이 걸리는 경우와 같이 udev 이벤트가 실행될 때 까지 지연이 발생할 수 있습니다. 이로 인해 커널이 장치를 감지할 때와 /dev/disk/by-*/ 이름을 사용할 수 있는 시점 사이에 지연이 발생할 수 있습니다.
- 규칙에 의해 호출된 blkid 와 같은 외부 프로그램은 짧은 기간 동안 장치를 열어 장치를 다른 용도로 액세스할 수 없게 만듭니다.
- /dev/disk/의 udev 메커니즘에서 관리하는 장치 이름은 주요 릴리스마다 변경될 수 있으므로 링크를 업데이트해야 합니다.

13.6. 영구 이름 지정 속성 나열

다음 절차에서는 비영구 스토리지 장치의 영구 이름 지정 속성을 찾는 방법을 설명합니다.

절차

- **UUID** 및 레이블 속성을 나열하려면 **lsblk** 유틸리티를 사용합니다.

```
$ lsblk --fs storage-device
```

예를 들면 다음과 같습니다.

예 13.4. 파일 시스템의 **UUID** 및 레이블 보기

```
$ lsblk --fs /dev/sda1

NAME FSTYPE LABEL UUID                                 MOUNTPOINT
sda1 xfs    Boot  afa5d5e3-9050-48c3-acc1-bb30095f3dc4 /boot
```

- **PARTUUID** 특성을 나열하려면 **--output +PARTUUID** 옵션과 함께 **lsblk** 유틸리티를 사용합니다.

```
$ lsblk --output +PARTUUID
```

예를 들면 다음과 같습니다.

예 13.5. 파티션의 **PARTUUID** 속성 보기

```
$ lsblk --output +PARTUUID /dev/sda1

NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT PARTUUID
sda1   8:1    0  512M  0 part /boot      4cd1448a-01
```

- **WWID** 특성을 나열하려면 **/dev/disk/by-id/** 디렉토리에 있는 심볼릭 링크 대상을 검사합니다. 예를 들면 다음과 같습니다.

예 13.6. 시스템에 있는 모든 스토리지 장치의 **WWID** 보기

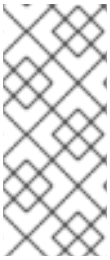
```
$ file /dev/disk/by-id/*

/dev/disk/by-id/ata-QEMU_HARDDISK_QM00001
symbolic link to ../../sda
/dev/disk/by-id/ata-QEMU_HARDDISK_QM00001-part1
symbolic link to ../../sda1
/dev/disk/by-id/ata-QEMU_HARDDISK_QM00001-part2
symbolic link to ../../sda2
/dev/disk/by-id/dm-name-rhel_rhel8-root
symbolic link to ../../dm-0
```

```
/dev/disk/by-id/dm-name-rhel_rhel8-swap
symbolic link to ../../dm-1
/dev/disk/by-id/dm-uuid-LVM-
QIWtEHtXGobe5bewlIUDivKOz5ofkgFhP0RMFsNyySVihqEl2cWWbR7MjXJolD6g
symbolic link to ../../dm-1
/dev/disk/by-id/dm-uuid-LVM-
QIWtEHtXGobe5bewlIUDivKOz5ofkgFhXqH2M45hD2H9nAf2qfWSrlRLhzfMyOKd
symbolic link to ../../dm-0
/dev/disk/by-id/lvm-pv-uuid-atlr2Y-vuMo-ueoH-CpMG-4JuH-AhEF-wu4QQm
symbolic link to ../../sda2
```

13.7. 영구 이름 지정 속성 수정

다음 절차에서는 파일 시스템의 **UUID** 또는 레이블 영구 이름 지정 속성을 변경하는 방법을 설명합니다.



참고

udev 특성을 변경하면 백그라운드에서 발생하며 시간이 오래 걸릴 수 있습니다. **udevadm settle** 명령은 변경 사항이 완전히 등록될 때까지 대기하므로 다음 명령이 새 특성을 올바르게 사용할 수 있습니다.

다음 명령에서 다음을 수행합니다.

- 설정할 **UUID**(예: **1cdfbc07-1c90-b5ec-f61943f5ea50**)로 **new-uuid** 를 바꿉니다. **uuidgen** 명령을 사용하여 **UUID**를 생성할 수 있습니다.
- **new-label** 을 레이블로 교체합니다(예: **backup_data**).

사전 요구 사항

- **XFS** 파일 시스템의 속성을 수정하는 경우 먼저 마운트 해제합니다.

절차

- **XFS** 파일 시스템의 **UUID** 또는 레이블 속성을 변경하려면 **xfs_admin** 유틸리티를 사용합니다.

```
# xfs_admin -U new-uuid -L new-label storage-device
# udevadm settle
```

- ***ext4,ext3*** *또는* ***ext2*** *파일 시스템의* ***UUID*** *또는 레이블 속성을 변경하려면* ***tune2fs*** *유틸리티를 사용합니다.*

```
# tune2fs -U new-uuid -L new-label storage-device
# udevadm settle
```

- *스왑 볼륨의* ***UUID*** *또는 레이블 속성을 변경하려면* ***swaplabel*** *유틸리티를 사용합니다.*

```
# swaplabel --uuid new-uuid --label new-label swap-device
# udevadm settle
```

14장. 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 번호: **ATA1.8.0SUNG MZNLN256(scsi)**

디스크 유형, 제조업체, 모델 번호 및 인터페이스입니다.

디스크 **/dev/sda: 256GB**

블록 장치 및 스토리지 용량의 파일 경로입니다.

파티션 테이블: **msdos**

디스크 레이블 유형입니다.

숫자

*파티션 번호입니다. 예를 들어, 마이너 번호 **1**이 있는 파티션은 **/dev/sda1** 에 해당합니다.*

시작 및 종료

파티션이 시작되고 종료되는 장치의 위치입니다.

유형

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

파일 시스템

*파일 시스템 유형입니다. 장치의 파일 시스템 필드에 값이 표시되지 않으면 파일 시스템 유형이 알 수 없음을 의미합니다. **parted** 유틸리티는 암호화된 장치에서 파일 시스템을 인식할 수 없습니다.*

플래그

*파티션에 설정된 플래그를 나열합니다. 사용 가능한 플래그는 **boot,root,swap,hidden,raid,lvm,** 또는 **lba** 입니다.*

추가 리소스

- ***parted(8)** 도움말 페이지.*

15장. PARTED로 디스크에 파티션 테이블 만들기

parted 유틸리티를 사용하여 파티션 테이블로 블록 장치를 보다 쉽게 포맷합니다.

주의

블록 장치를 파티션 테이블로 포맷하면 장치에 저장된 모든 데이터가 삭제됩니다.

절차

1. 대화형 **parted** 쉘을 시작합니다.

```
# parted block-device
```

2. 장치에 파티션 테이블이 이미 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.
  - **1500**용 **MS DOS**
  - **GPT**의 경우 **GPT**

예 **15.1. 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.*

16장. 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. 현재 파티션 테이블을 보고 사용 가능한 공간이 충분한지 확인합니다.

```
# (parted) print
```

- 사용 가능한 공간이 충분하지 않은 경우 파티션의 크기를 조정합니다.
- 파티션 테이블에서 다음을 확인합니다.
  - 새 파티션의 시작 및 끝점입니다.
  - **EgressIP**에서 사용해야 하는 파티션 유형은 무엇입니까.

3.

새 파티션을 만듭니다.

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

- ***part-type*** *을 기본,논리 또는 확장으로 바꿉니다. 이 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 ***16.1.*** *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 파티션 테이블에 올바른 파티션 유형, 파일 시스템 유형 및 크기가 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.*
- ***parted**를 사용하여 디스크에서 파티션 테이블 만들기*
- ***parted**로 파티션 크기 조정*

17장. 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
```

   - 블록 장치를 파티션을 제거하려는 장치의 경로로 바꿉니다(예: **/dev/sda**).

2. 제거할 파티션의 마이너 번호를 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 선언하는 행을 찾아 파일에서 제거합니다.*

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

```
# systemctl daemon-reload
```

9. *스왑 파티션을 삭제하거나 **LVM**을 제거한 경우 커널 명령줄에서 파티션에 대한 모든 참조를 제거하십시오.*

   a. *활성 커널 옵션을 나열하고 옵션이 제거된 파티션을 참조하는지 확인합니다.*

```
# grubby --info=ALL
```

   b. *삭제된 파티션을 참조하는 커널 옵션을 제거합니다.*

```
# grubby --update-kernel=ALL --remove-args="option"
```

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

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

추가 리소스

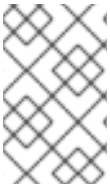
- ***parted(8)** 도움말 페이지*

18장. 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 조정 파티션이 파티션 테이블에 올바른 크기인지 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.

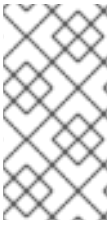
추가 리소스

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

19장. 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.



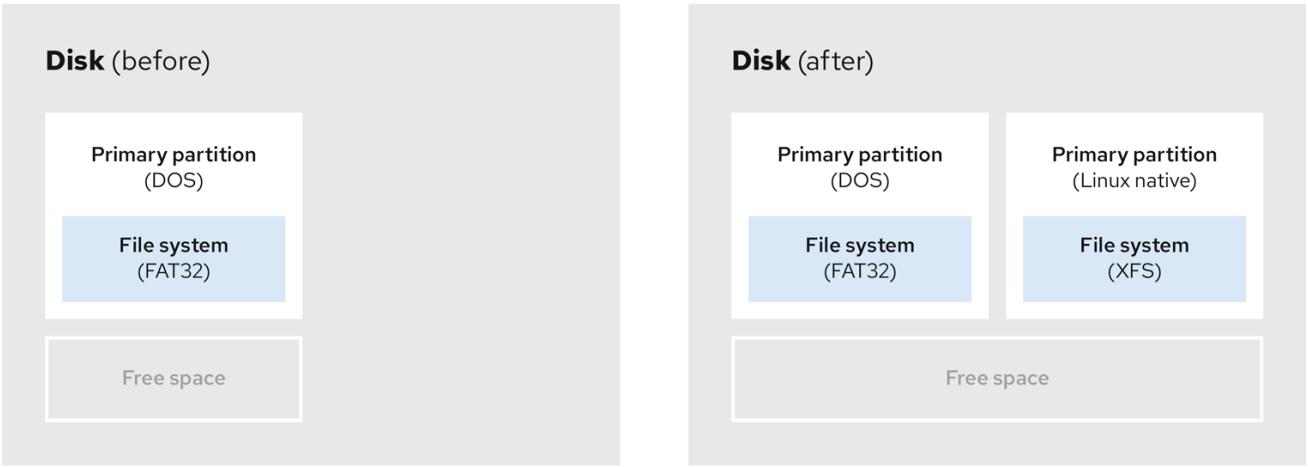
참고

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

19.1. 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 19.1. 파티션되지 않은 여유 공간이 있는 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.

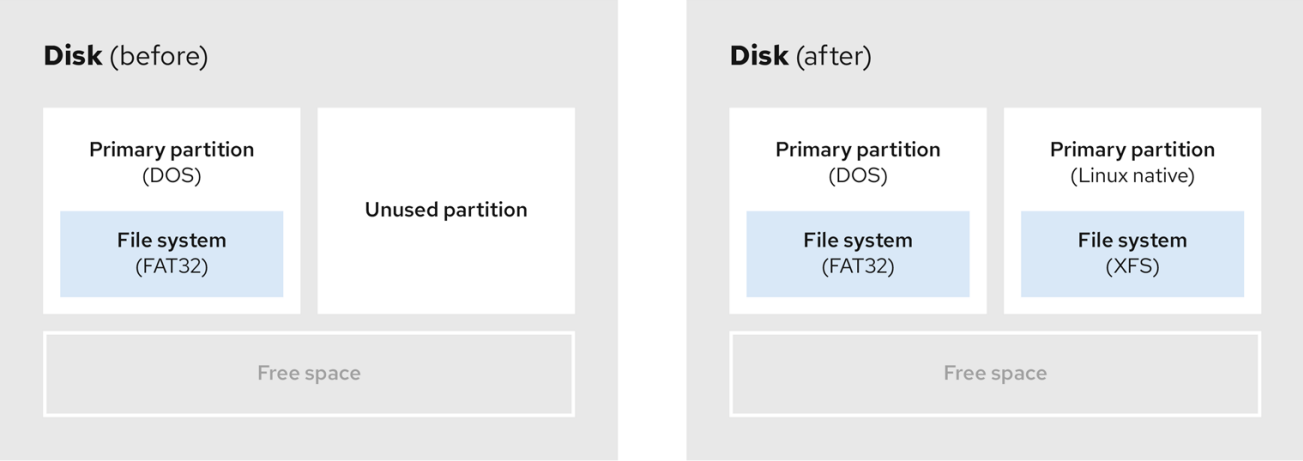
사용되지 않은 하드 디스크도 이 분류에 속합니다. 유일한 차이점은 모든 공간이 정의된 파티션의 일부가 아니라는 것입니다.

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

19.2. 사용되지 않는 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 **19.2.** 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.*

19.3. 활성 파티션에서 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.

19.3.1. 안전하지 않은 재파티션

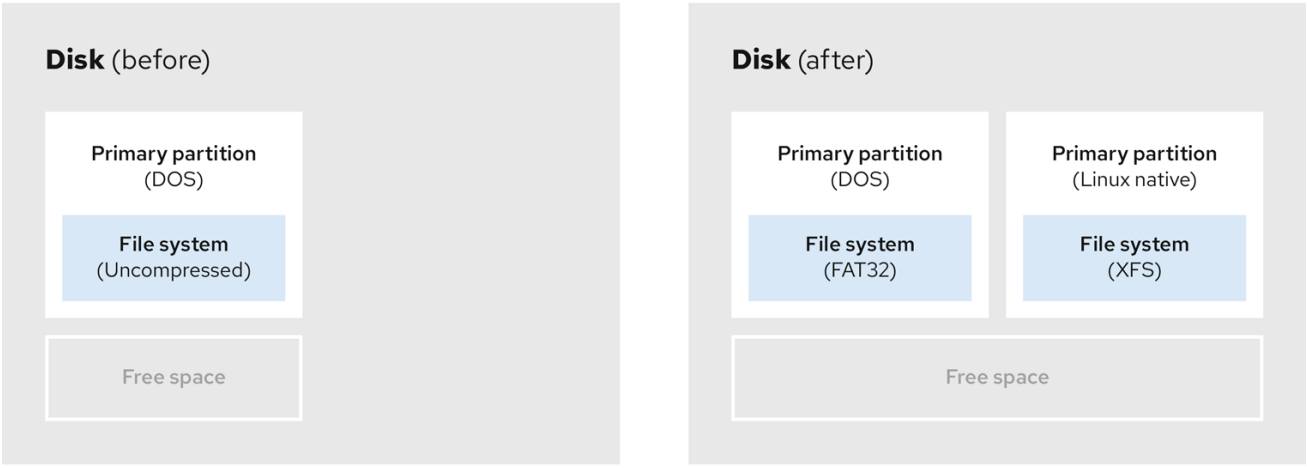
안전하지 않은 재파티션은 하드 드라이브에서 파티션을 제거하고 대신 작은 여러 파티션을 만듭니다. 이 방법을 사용하면 전체 내용이 삭제되므로 원래 파티션에서 필요한 모든 데이터를 백업하십시오.

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 19.3. 디스크에 대한 안전하지 않은 재파티 파티션 작업



주의

이 방법은 원래 파티션에 이전에 저장된 모든 데이터를 삭제합니다.

19.3.2. 강제 다시 분할되지 않은 파티션 지정

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

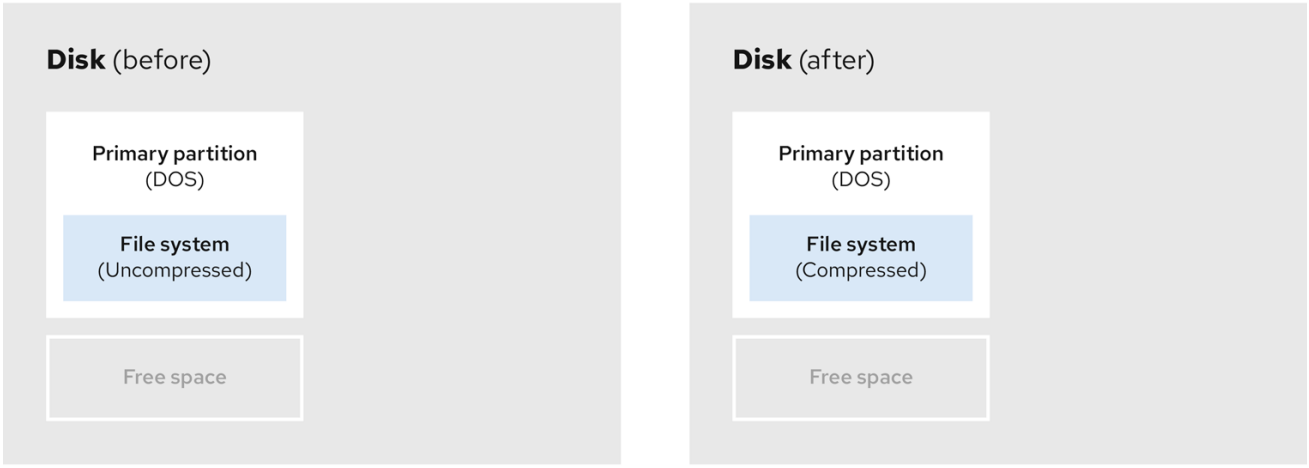
다음은 거부된 복원을 시작하는 데 도움이 될 수 있는 메서드 목록입니다.

- *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 19.4. 디스크의 데이터 압축



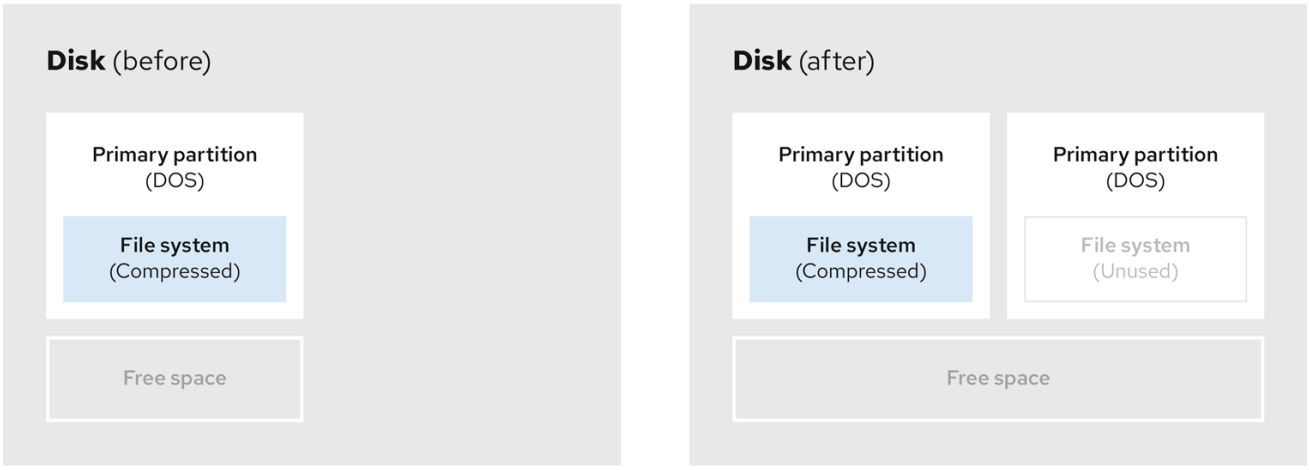
가능한 데이터 손실을 방지하려면 압축 프로세스를 계속하기 전에 백업을 만듭니다.

- *기존 파티션 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 19.5. 디스크의 파티션 크기 조정

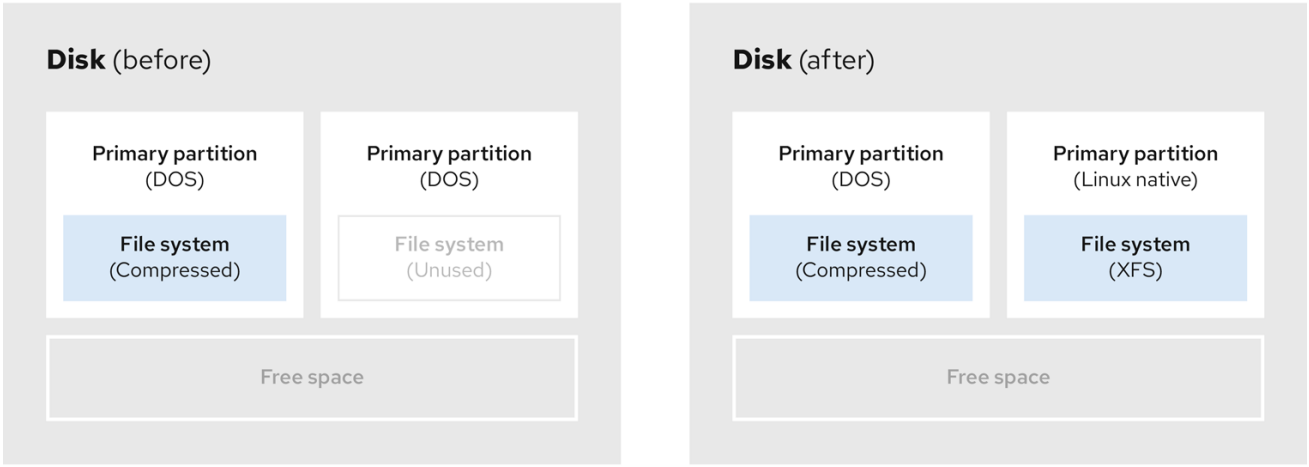


- *선택 사항: 새 파티션 생성*

*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 **19.6.** 최종 파티션 구성이 있는 디스크*



20장. XFS 시작하기

다음은 **XFS** 파일 시스템을 생성하고 유지 관리하는 방법에 대한 개요입니다.

20.1. XFS 파일 시스템

XFS는 단일 호스트에서 매우 큰 파일 및 파일 시스템을 지원하는 확장성이 뛰어난 고성능, 견고하며 성숙한 **64**비트 저널링 파일 시스템입니다. **Red Hat Enterprise Linux 9**의 기본 파일 시스템입니다. **XFS**는 원래 **SGI**에 의해 **Subversions**에서 개발되었으며 매우 큰 서버 및 스토리지 어레이에서 실행하는 오랜 기록을 보유하고 있습니다.

XFS의 기능은 다음과 같습니다.

신뢰성

- 메타데이터 저널링 - 시스템을 다시 시작하고 파일 시스템을 다시 시작할 때 다시 마운트할 수 있는 파일 시스템 작업 레코드를 유지하여 시스템 충돌 후 파일 시스템 무결성을 보장합니다.
- 광범위한 런타임 메타데이터 일관성 확인
- 확장 가능하고 빠른 복구 유틸리티
- 할당량 저널링. 이렇게 하면 충돌 후 긴 할당량 일관성 점검이 필요하지 않습니다.

확장 및 성능

- 지원되는 파일 시스템 크기는 최대 **1024TiB**
- 다수의 동시 작업을 지원하는 기능
- 여유 공간 관리의 확장성을 위한 **B-tree** 인덱싱
- 정교한 메타데이터 읽기 알고리즘

- *비디오 워크로드 스트리밍 최적화*

할당 체계

- *범위 기반 할당*
- *스트라이프 인식 할당 정책*
- *지연된 할당*
- *공간 사전 할당*
- *동적으로 할당된 **inode***

기타 기능

- ***reflink** 기반 파일 복사*
- *긴밀하게 통합된 백업 및 복원 유틸리티*
- *온라인 조각 모음*
- *온라인 파일 시스템 확장*
- *포괄적인 진단 기능*
- *확장 속성(**xattr**). 이를 통해 시스템은 파일당 여러 개의 추가 이름/값 쌍을 연결할 수 있습니다.*
- *프로젝트 또는 디렉터리 할당량입니다. 이렇게 하면 디렉터리 트리에 대한 할당량 제한을 사용할 수 있습니다.*

- 2초 타임 스탬프

성능 특성

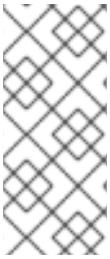
XFS는 엔터프라이즈 워크로드가 있는 대규모 시스템에서 고성능을 보유하고 있습니다. 대규모 시스템은 비교적 많은 CPU, 여러 HBA 및 외부 디스크 어레이 연결을 사용하는 시스템입니다. XFS는 멀티 스레드 병렬 I/O 워크로드가 있는 소규모 시스템에서도 제대로 작동합니다.

XFS는 단일 스레드에 대해 상대적으로 낮은 성능을 제공합니다. 예를 들어 단일 스레드에서 많은 수의 작은 파일을 생성하거나 삭제하는 워크로드와 같습니다.

20.2. EXT4 및 XFS와 함께 사용되는 툴 비교

이 섹션에서는 ext4 및 XFS 파일 시스템에서 일반적인 작업을 수행하는 데 사용할 툴을 비교합니다.

Task	ext4	XFS
파일 시스템 생성	**mkfs.ext4**	**mkfs.xfs**
파일 시스템 확인	**e2fsck**	**xfs_repair**
파일 시스템 크기 조정	**resize2fs**	**xfs_growfs**
파일 시스템의 이미지 저장	**e2image**	**xfs_metadump** 및 **xfs_mdrestore**
파일 시스템의 레이블 또는 튜닝	**tune2fs**	**xfs_admin**
파일 시스템 백업	**tar** 및 **rsync**	**xfsdump** 및 **xfsrestore**
할당량 관리	**할당량**	**xfs_quota**
파일 매핑	**filefrag**	**xfs_bmap**



참고

네트워크를 통한 백업에 대한 전체 클라이언트-서버 솔루션을 원한다면 RHEL 9에서 사용할 수 있는 bacula 백업 유틸리티를 사용할 수 있습니다. Bacula에 대한 자세한 내용은 Bacula 백업 솔루션을 참조하십시오.

21장. XFS 파일 시스템 생성

시스템 관리자는 블록 장치에 **XFS** 파일 시스템을 생성하여 파일 및 디렉터리를 저장할 수 있습니다.

21.1. MKFS.XFS를 사용하여 XFS 파일 시스템 생성

다음 절차에서는 블록 장치에 **XFS** 파일 시스템을 생성하는 방법을 설명합니다.

절차

1. 파일 시스템을 생성하려면 다음을 수행합니다.

- 장치가 일반 파티션, **LVM** 볼륨, **MD** 볼륨, 디스크 또는 유사한 장치인 경우 다음 명령을 사용합니다.

```
# mkfs.xfs block-device
```

  - **block-device** 를 블록 장치의 경로로 바꿉니다. 예: **/dev/sdb1**,**/dev/disk/by-uuid/05e99ec8-def1-4a5e-8a9d-5945339ceb2a**, 또는 **/dev/my-volgroup/my-lv**.

  - 일반적으로 기본 옵션은 일반적으로 사용하기에 적합합니다.

  - 기존 파일 시스템이 포함된 블록 장치에서 **mkfs.xfs** 를 사용하는 경우 **-f** 옵션을 추가하여 해당 파일 시스템을 덮어씁니다.

- 하드웨어 **RAID** 장치에 파일 시스템을 생성하려면 시스템이 장치의 스트라이프형을 올바르게 탐지하는지 확인합니다.

  - 스트라이프 기하형 정보가 올바르면 추가 옵션이 필요하지 않습니다. 파일 시스템을 생성합니다.

```
# mkfs.xfs block-device
```

  - 정보가 올바르지 않으면 **-d** 옵션의 **su** 및 **sw** 매개변수를 사용하여 스트라이프 기하를 수동으로 지정합니다. **su** 매개 변수는 **RAID** 청크 크기를 지정하고 **sw** 매개 변수는

RAID 장치의 데이터 디스크 수를 지정합니다.

예를 들면 다음과 같습니다.

```
# mkfs.xfs -d su=64k,sw=4 /dev/sda3
```

2. 다음 명령을 사용하여 시스템이 새 장치 노드를 등록할 때까지 기다립니다.

```
# udevadm settle
```

추가 리소스

- **mkfs.xfs(8)** 도움말 페이지.

22장. RHEL 시스템 역할을 사용하여 블록 장치에서 XFS 파일 시스템 생성

이 섹션에서는 스토리지 역할을 사용하여 여러 대상 시스템의 블록 장치에 **XFS** 파일 시스템을 생성하는 방법을 설명합니다.

사전 요구 사항

- 스토리지 역할을 사용하는 **Ansible** 플레이북이 있습니다.

22.1. 예제 ANSIBLE PLAYBOOK 블록 장치에 XFS 파일 시스템을 생성

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 기본 매개 변수를 사용하여 블록 장치에 **XFS** 파일 시스템을 생성하도록 **storage** 역할을 적용합니다.

주의

스토리지 역할은 분할되지 않은 전체 디스크 또는 논리 볼륨(LV)에서만 파일 시스템을 생성할 수 있습니다. 파티션에 파일 시스템을 만들 수 없습니다.

예 22.1. /dev/sdb에서 XFS를 생성하는 플레이북

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
  roles:
    - rhel-system-roles.storage
```

- 볼륨 이름(예의**barefs**)은 현재 임의로 사용할 수 있습니다. 스토리지 역할은 **disks:** 속성 아래에 나열된 디스크 장치에서 볼륨을 식별합니다.

- **XFS**는 **RHEL 9**의 기본 파일 시스템이므로 **fs_type: xfs** 행을 생략할 수 있습니다.

- *LV에 파일 시스템을 생성하려면 **enclosing** 볼륨 그룹을 포함하여 **disks:** 속성 아래에 LVM 설정을 제공합니다. 자세한 내용은 Example Ansible Playbook to manage logical volumes 에서 참조하십시오.*

  LV 장치의 경로를 제공하지 마십시오.

추가 리소스

- */usr/share/ansible/roles/rhel-system-roles.storage/README.md 파일.*

22.2. 추가 리소스

- *스토리지 역할 소개*

23장. XFS 파일 시스템 백업

시스템 관리자는 xfsdump 를 사용하여 파일 또는 테이프에 XFS 파일 시스템을 백업할 수 있습니다. 이는 간단한 백업 메커니즘을 제공합니다.

23.1. XFS 백업 기능

이 섹션에서는 xfsdump 유틸리티를 사용하여 XFS 파일 시스템을 백업하는 주요 개념과 기능을 설명합니다.

xfsdump 유틸리티를 사용하여 다음을 수행할 수 있습니다.

- 일반 파일 이미지로 백업을 수행합니다.

  일반 파일에 하나의 백업만 쓸 수 있습니다.

- Backape 드라이브로 백업을 수행합니다.

  또한 xfsdump 유틸리티를 사용하면 동일한 테이프에 여러 백업을 작성할 수 있습니다. 백업은 여러 개의 테이프에 걸쳐 있을 수 있습니다.

  여러 파일 시스템을 단일 minion 장치에 백업하려면 이미 XFS 백업이 포함된 테이프에 백업을 쓰기만 하면 됩니다. 그러면 새 백업이 이전 백업에 추가됩니다. 기본적으로 xfsdump 는 기존 백업을 덮어쓰지 않습니다.

- 증분 백업을 생성합니다.

  xfsdump 유틸리티는 덤프 수준을 사용하여 다른 백업이 상대적 기본 백업을 결정합니다. 0에서 9까지의 숫자는 덤프 수준 증가를 나타냅니다. 증분 백업에서는 하위 수준의 마지막 덤프 이후 변경된 파일만 백업합니다.

  - 전체 백업을 수행하려면 파일 시스템에서 수준 0 덤프를 수행합니다.
  - 수준 1 덤프는 전체 백업 후 첫 번째 증분 백업입니다. 다음 증분 백업은 마지막 레벨 1 덤프 이후 변경된 파일만 백업하는 수준 2입니다.

- 크기, 하위 트리 또는 **inode** 플래그를 사용하여 백업에서 파일을 필터링하여 제외합니다.

추가 리소스

- **xfsdump(8)** 도움말 페이지.

23.2. XFSDUMP를 사용하여 XFS 파일 시스템 백업

다음 절차에서는 **XFS** 파일 시스템의 콘텐츠를 파일 또는 테이프에 백업하는 방법을 설명합니다.

사전 요구 사항

- 백업할 수 있는 **XFS** 파일 시스템입니다.
- 백업을 저장할 수 있는 또 다른 파일 시스템 또는핑 드라이브입니다.

절차

- 다음 명령을 사용하여 **XFS** 파일 시스템을 백업합니다.

```
# xfsdump -l level [-L label] \
        -f backup-destination path-to-xfs-filesystem
```

  - 수준을 백업의 덤프 수준으로 바꿉니다. **0** 을 사용하여 전체 백업을 수행하거나 **1~9** 를 사용하여 연속 증분 백업을 수행합니다.
  - **backup-destination** 을 백업 저장 경로로 교체합니다. 대상은 일반 파일, 백업 드라이브 또는 원격 백업 장치일 수 있습니다.**The destination can be a regular file, a tape drive, or a remote tape device.** 예를 들어, 백피드 드라이브의 경우 **/backup-files/Data.xfsdump for a file** 또는 **/dev/st0** 입니다.
  - **path-to-xfs-filesystem** 을 백업하려는 **XFS** 파일 시스템의 마운트 지점으로 교체합니다. 예를 들면 **/mnt/data/** 입니다. 파일 시스템을 마운트해야 합니다.
  -

여러 파일 시스템을 백업하고 단일 명령식 장치에 저장할 때 **-L** 레이블 옵션을 사용하여 각 백업에 세션 레이블을 추가하여 복원 시 쉽게 식별할 수 있도록 합니다. 레이블을 백업의 모든 이름으로 교체합니다(예: **backup_data**).

예 23.1. 여러 **XFS** 파일 시스템 백업

- **/boot/** 및 **/data/** 디렉토리에 마운트된 **XFS** 파일 시스템의 콘텐츠를 백업하고 이를 **/backup-files/** 디렉터리에 파일로 저장하려면 다음을 실행합니다.

```
# xfsdump -l 0 -f /backup-files/boot.xfsdump /boot
# xfsdump -l 0 -f /backup-files/data.xfsdump /data
```

- 단일 **Dockerfile** 장치에서 여러 파일 시스템을 백업하려면 **-L** 레이블 옵션을 사용하여 각 백업에 세션 레이블을 추가합니다.

```
# xfsdump -l 0 -L "backup_boot" -f /dev/st0 /boot
# xfsdump -l 0 -L "backup_data" -f /dev/st0 /data
```

추가 리소스

- **xfsdump(8)** 도움말 페이지.

23.3. 추가 리소스

- **xfsdump(8)** 도움말 페이지.

24장. 백업에서 XFS 파일 시스템 복원

시스템 관리자는 **xfsrestore** 유틸리티를 사용하여 **xfsdump** 유틸리티로 생성된 **XFS** 백업을 복원하고 파일 또는 테이프에 저장할 수 있습니다.

24.1. 백업에서 XFS 복원 기능

xfsrestore 유틸리티는 **xfsdump** 로 생성된 백업에서 파일 시스템을 복원합니다. **xfsrestore** 유틸리티에는 다음 두 가지 모드가 있습니다.

- 간단한 모드를 사용하면 전체 파일 시스템을 수준 **0** 덤프에서 복원할 수 있습니다. 이는 기본값 모드입니다.
- 누적 모드를 사용하면 파일 시스템을 증분 백업에서 복구할 수 있습니다. 즉, 수준 **1**에서 수준 **9**까지입니다.

고유한 세션 **ID** 또는 세션 레이블 은 각 백업을 식별합니다. 여러 백업이 포함된 백업에서 백업을 복원하려면 해당 세션 **ID** 또는 레이블이 필요합니다.

백업에서 특정 파일을 추출, 추가 또는 삭제하려면 **xfsrestore** 대화형 모드를 입력합니다. 대화형 모드는 백업 파일을 조작하는 명령 집합을 제공합니다.

추가 리소스

- **xfsrestore(8)** 도움말 페이지.

24.2. XFSRESTORE를 사용하여 백업에서 XFS 파일 시스템 복원

다음 절차에서는 파일 또는 테이프 백업에서 **XFS** 파일 시스템의 콘텐츠를 복원하는 방법을 설명합니다.

사전 요구 사항

- **XFS** 파일 시스템 백업에 설명된 대로 **XFS** 파일 시스템의 파일 또는 테이프 백업입니다.

- 백업을 복원할 수 있는 스토리지 장치입니다.

절차

- 백업을 복원하는 명령은 전체 백업 또는 증분 백업에서 복원하는지 아니면 단일 명령식 장치에서 여러 백업을 복원할지 여부에 따라 다릅니다.

```
# xfsrestore [-r] [-S session-id] [-L session-label] [-i]
        -f backup-location restoration-path
```

  - **backup-location** 을 백업 위치로 교체합니다. 이는 일반 파일, 백업 드라이브 또는 원격 백업 장치일 수 있습니다.**This can be a regular file, a tape drive, or a remote tape device.** 예를 들어, 백피드 드라이브의 경우 **/backup-files/Data.xfsdump for a file** 또는 **/dev/st0** 입니다.

  - **restore -path** 를 파일 시스템을 복원하려는 디렉터리의 경로로 교체합니다. 예를 들면 **/mnt/data/** 입니다.

  - 증분(레벨 **1** 수준 **9**) 백업에서 파일 시스템을 복원하려면 **-r** 옵션을 추가합니다.

  - 여러 백업이 포함된 기존 장치에서 백업을 복원하려면 **-S** 또는 **-L** 옵션을 사용하여 백업을 지정합니다.**To restore a backup from a tape device that contains multiple backups, specify the backup using the -S or -L options.**

    -S 옵션을 사용하면 세션 **ID**로 백업을 선택할 수 있지만 **-L** 옵션을 사용하면 세션 레이블에 따라 선택할 수 있습니다. 세션 **ID** 및 세션 레이블을 가져오려면 **xfsrestore -I** 명령을 사용합니다.

    session-id 를 백업 세션 **ID**로 교체합니다. 예를 들어 **b74a3586-e52e-4a4a-8775-c3334fa8ea2c. session-label** 을 백업의 **session** 레이블로 바꿉니다. 예: **my_backup_session_label.**

  - **xfsrestore** 를 대화형으로 사용하려면 **-i** 옵션을 사용합니다.

    대화형 대화는 **xfsrestore** 가 지정된 장치 읽기를 완료한 후에 시작됩니다. 대화형 **xfsrestore** 쉘에서 사용 가능한 명령에는 **cd,ls,add,delete, extract;** 전체 명령 목록을 보려면 **help** 명령을 사용합니다.

예 24.1. 여러 XFS 파일 시스템 복원

- **XFS** 백업 파일을 복원하고 콘텐츠를 **/mnt/** 아래의 디렉터리에 저장하려면 다음을 수행합니다.

```
# xfsrestore -f /backup-files/boot.xfsdump /mnt/boot/
# xfsrestore -f /backup-files/data.xfsdump /mnt/data/
```

- 여러 개의 백업이 포함된 녹화된 장치에서 복원하려면 세션 레이블 또는 세션 **ID**로 각 백업을 지정합니다.

```
# xfsrestore -L "backup_boot" -f /dev/st0 /mnt/boot/
# xfsrestore -S "45e9af35-efd2-4244-87bc-4762e476cbab" \
        -f /dev/st0 /mnt/data/
```

추가 리소스

- **xfsrestore(8)** 도움말 페이지.

24.3. 테이프에서 XFS 백업을 복원할 때 정보 메시지

여러 파일 시스템에서 백업으로 백업을 복원할 때 **xfsrestore** 유틸리티에서 메시지를 발행할 수 있습니다. 이 메시지는 **xfsrestore** 에서 각 백업을 순서대로 순서대로 검사할 때 요청된 백업의 일치 여부를 알려줍니다. 예를 들면 다음과 같습니다.

```
xfsrestore: preparing drive
xfsrestore: examining media file 0
xfsrestore: inventory session uuid (8590224e-3c93-469c-a311-fc8f23029b2a) does not match the
media header's session uuid (7eda9f86-f1e9-4dfd-b1d4-c50467912408)
xfsrestore: examining media file 1
xfsrestore: inventory session uuid (8590224e-3c93-469c-a311-fc8f23029b2a) does not match the
media header's session uuid (7eda9f86-f1e9-4dfd-b1d4-c50467912408)
[...]
```

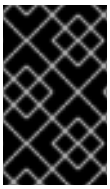
일치하는 백업이 발견될 때까지 정보 메시지가 계속 표시됩니다.

24.4. 추가 리소스

- **xfsrestore(8)** 도움말 페이지.

25장. XFS 파일 시스템의 크기 늘리기

시스템 관리자는 더 큰 스토리지 용량을 사용하기 위해 **XFS** 파일 시스템의 크기를 늘릴 수 있습니다.



중요

현재 **XFS** 파일 시스템의 크기를 줄일 수 없습니다.

25.1. XFS_GROWFS를 사용하여 XFS 파일 시스템의 크기 증가

다음 절차에서는 **xfs_growfs** 유틸리티를 사용하여 **XFS** 파일 시스템을 확장하는 방법을 설명합니다.

사전 요구 사항

- 기본 블록 장치가 나중에 크기 조정 파일 시스템을 유지하는 적절한 크기인지 확인합니다. 영향을 받는 블록 장치에 적절한 크기 조정 방법을 사용합니다.
- **XFS** 파일 시스템을 마운트합니다.

절차

- **XFS** 파일 시스템이 마운트되지만 **xfs_growfs** 유틸리티를 사용하여 크기를 늘립니다.

```
# xfs_growfs file-system -D new-size
```

  - 파일 시스템을 **XFS** 파일 시스템의 마운트 지점으로 바꿉니다.
  - **d** 옵션을 사용하여 **new-size** 를 파일 시스템 블록 수에 지정된 파일 시스템의 원하는 새 크기로 바꿉니다.

    지정된 **XFS** 파일 시스템의 **kB**에서 블록 크기를 확인하려면 **xfs_info** 유틸리티를 사용합니다.

```
# xfs_info block-device
...
```

```
data     =                       bsize=4096
...
```

- **d** 옵션이 없으면 **xfs_growfs** 는 파일 시스템을 기본 장치에서 지원하는 최대 크기로 확장합니다.

추가 리소스

- **xfs_growfs(8)** 도움말 페이지.

26장. XFS 오류 동작 구성

XFS 파일 시스템이 다른 I/O 오류가 발생할 때 작동하는 방식을 구성할 수 있습니다.

26.1. XFS에서 구성 가능한 오류 처리

XFS 파일 시스템은 I/O 작업 중에 오류가 발생하면 다음 방법 중 하나로 응답합니다.

- XFS는 작업이 성공하거나 XFS가 설정된 제한에 도달할 때까지 I/O 작업을 반복적으로 다시 시도합니다.

  제한은 최대 재시도 횟수 또는 재시도의 최대 시간을 기반으로 합니다.

- XFS는 영구적으로 오류를 고려하고 파일 시스템에서 작업을 중지합니다.

XFS가 다음 오류 조건에 응답하는 방법을 구성할 수 있습니다.

EIO

읽기 또는 쓰기 시 오류

ENOSPC

장치에 남은 공간 없음

ENODEV

장치를 찾을 수 없음

XFS가 오류를 영구적으로 간주할 때까지 최대 재시도 횟수와 최대 시간(초)을 설정할 수 있습니다. XFS는 제한 중 하나에 도달하면 작업 재시도를 중지합니다.

XFS를 구성하여 파일 시스템을 마운트 해제할 때 XFS는 다른 구성과 관계없이 즉시 재시도를 취소할 수 있습니다. 이 구성을 사용하면 영구 오류에도 불구하고 마운트 해제 작업이 성공할 수 있습니다.

기본 동작

각 **XFS** 오류 조건에 대한 기본 동작은 오류 컨텍스트에 따라 다릅니다. **ENODEV** 와 같은 일부 **XFS** 오류는 재시도 횟수에 관계없이 치명적이고 복구할 수 없는 것으로 간주됩니다. 기본 재시도 제한은 **0**입니다.

26.2. 특정 및 정의되지 않은 XFS 오류 조건에 대한 구성 파일

다음 디렉터리는 다양한 오류 조건에 대해 **XFS** 오류 동작을 제어하는 구성 파일을 저장합니다.

/sys/fs/xfs/device/error/metadata/EIO/

EIO 오류 조건의 경우

/sys/fs/xfs/device/error/metadata/ENODEV/

ENODEV 오류 조건의 경우

/sys/fs/xfs/device/error/metadata/ENOSPC/

ENOSPC 오류 조건의 경우

/sys/fs/xfs/device/error/default/

정의되지 않은 기타 모든 오류 조건에 대한 일반 구성

각 디렉터리에는 재시도 제한을 구성하기 위한 다음 구성 파일이 포함되어 있습니다.

max_retries

XFS에서 작업을 재시도하는 최대 횟수를 제어합니다.

retry_timeout_seconds

XFS가 작업 재시도를 중지한 후 시간 제한을 초 단위로 지정합니다.

26.3. 특정 조건에 대한 XFS 동작 설정

다음 절차에서는 **XFS**가 특정 오류 상태에 대응하는 방법을 구성합니다.

절차

- 최대 재시도 횟수, 재시도 시간 제한 또는 둘 다를 설정합니다.
  - 최대 재시도 횟수를 설정하려면 **max_retries** 파일에 원하는 번호를 작성합니다.

```
# echo value > /sys/fs/xfs/device/error/metadata/condition/max_retries
```

  - 시간 제한을 설정하려면 **retry_timeout_seconds** 파일에 원하는 초 수를 씁니다.

```
# echo value > /sys/fs/xfs/device/error/metadata/condition/retry_timeout_second
```

value 는 **-1**과 **C** 부호 있는 정수 유형의 가능한 최대값 사이의 숫자입니다. **64**비트 **Linux**에서 **2147483647**입니다.

두 제한 모두에서 값 **-1** 은 연속 재시도에 사용되며 **0** 은 즉시 중지됩니다.

device 는 **/dev/** 디렉토리에 있는 장치 이름입니다(예: **sda**).

26.4. 정의되지 않은 조건에 대한 XFS 동작 설정

이 절차에서는 **XFS**가 공통 구성을 공유하는 모든 정의되지 않은 오류 조건에 반응하는 방법을 구성합니다.

절차

- 최대 재시도 횟수, 재시도 시간 제한 또는 둘 다를 설정합니다.
  - 최대 재시도 횟수를 설정하려면 **max_retries** 파일에 원하는 번호를 작성합니다.

```
# echo value > /sys/fs/xfs/device/error/metadata/default/max_retries
```

  - 시간 제한을 설정하려면 **retry_timeout_seconds** 파일에 원하는 초 수를 씁니다.

```
# echo value > /sys/fs/xfs/device/error/metadata/default/retry_timeout_seconds
```

value 는 *-1*과 *C* 부호 있는 정수 유형의 가능한 최대값 사이의 숫자입니다. *64*비트 *Linux*에서 *2147483647*입니다.

두 제한 모두에서 값 -1 은 연속 재시도에 사용되며 0 은 즉시 중지됩니다.

device 는 */dev/* 디렉토리에 있는 장치 이름입니다(예: *sda*).

26.5. XFS 마운트 해제 동작 설정

다음 절차에서는 XFS가 파일 시스템을 마운트 해제할 때 오류 상태에 대응하는 방법을 구성합니다.

파일 시스템에서 fail_at_unmount 옵션을 설정하면 마운트 해제 중에 다른 모든 오류 구성을 재정의하고 I/O 작업을 다시 시도하지 않고 파일 시스템을 즉시 마운트 해제합니다. 이를 통해 영구 오류 발생 시에도 마운트 해제 작업을 수행할 수 있습니다.

주의

마운트 해제 프로세스에서 해당 파일 시스템의 sysfs 인터페이스에서 구성 파일을 제거하므로 마운트 해제 프로세스가 시작된 후에는 fail_at_unmount 값을 변경할 수 없습니다. 파일 시스템의 마운트 해제를 시작하기 전에 마운트 해제 동작을 구성해야 합니다.

절차

- *fail_at_unmount 옵션을 활성화 또는 비활성화합니다.*
  - *파일 시스템이 마운트 해제될 때 모든 작업을 다시 시도하려면 옵션을 활성화합니다.*

```
# echo 1 > /sys/fs/xfs/device/error/fail_at_unmount
```

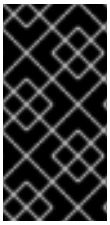
  - *파일 시스템이 마운트 해제될 때 max_retries 및 retry_timeout_seconds 재시도 제한을 유지하려면 옵션을 비활성화합니다.*

```
# echo 0 > /sys/fs/xfs/device/error/fail_at_unmount
```

***device** 는 **/dev/** 디렉토리에 있는 장치 이름입니다(예: **sda**).*

27장. 파일 시스템 검사 및 복구

RHEL은 파일 시스템을 확인하고 복구할 수 있는 파일 시스템 관리 유틸리티를 제공합니다. 이러한 툴을 fsck 도구라고 합니다. 여기서 fsck 는 파일 시스템 검사 의 단축된 버전입니다. 대부분의 경우 이러한 유틸리티는 시스템을 부팅하는 동안 자동으로 실행되지만 필요한 경우 수동으로 호출할 수도 있습니다.



중요

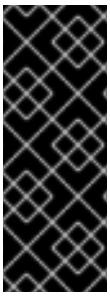
파일 시스템 검사기는 파일 시스템 전체에서 메타데이터 일관성만 보장합니다. 파일 시스템에 포함된 실제 데이터에 대한 인식이 없으며 데이터 복구 도구가 아닙니다.

27.1. 파일 시스템 확인이 필요한 시나리오

관련 fsck 툴은 다음 중 하나라도 발생하는 경우 시스템을 확인하는 데 사용할 수 있습니다.

- 시스템을 부팅할 수 없음
- 특정 디스크의 파일이 손상됨
- 불일치로 인해 파일 시스템을 종료하거나 읽기 전용으로 변경
- 파일 시스템의 파일에 액세스할 수 없습니다.

파일 시스템 불일치는 하드웨어 오류, 스토리지 관리 오류 및 소프트웨어 버그로 제한되지는 않지만 다양한 이유로 발생할 수 있습니다.

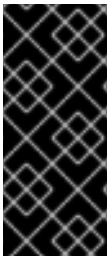


중요

파일 시스템 점검 도구는 하드웨어 문제를 복구할 수 없습니다. 복구가 성공적으로 작동하는 경우 파일 시스템을 완전히 읽고 쓸 수 있어야 합니다. 하드웨어 오류로 인해 파일 시스템이 손상된 경우 먼저 파일 시스템을 좋은 디스크로 이동해야 합니다(예: dd(8) 유틸리티).

저널링 파일 시스템의 경우 일반적으로 부팅 시 필요한 경우 저널을 재생하는 데 필요한 모든 작업은 매우 짧은 작업입니다.

그러나 파일 시스템 불일치 또는 손상이 발생하는 경우 파일 시스템 검사기를 사용하여 파일 시스템을 복구해야 합니다.



중요

/etc/fstab 의 여섯 번째 필드를 **0** 으로 설정하여 부팅 시 파일 시스템 점검을 비활성화할 수 있습니다. 그러나 **Red Hat**은 부팅시 **fsck (예: 매우 큰 또는 원격 파일 시스템)**에 문제가 있는 경우를 제외하고 그렇게 하지 않는 것이 좋습니다.

추가 리소스

- **fstab(5)** 도움말 페이지.
- **fsck(8)** 도움말 페이지.
- **DD(8)** 도움말 페이지.

27.2. FSCK 실행의 잠재적인 부작용이 발생할 수 있습니다.

일반적으로 파일 시스템 검사 및 복구 도구를 실행하면 발견된 불일치 중 적어도 일부를 자동으로 복구할 수 있습니다. 경우에 따라 다음과 같은 문제가 발생할 수 있습니다.

- 손상된 **inode** 또는 디렉터리를 복구할 수 없는 경우 삭제할 수 있습니다.
- 파일 시스템에 대한 중요한 변경이 발생할 수 있습니다.

예기치 않거나 바람직하지 않은 변경 사항이 영구적으로 수행되지 않도록 하려면 절차에 설명된 예방 단계를 따르십시오.

27.3. XFS의 오류 처리 메커니즘

이 섹션에서는 **XFS**가 파일 시스템의 다양한 종류의 오류를 처리하는 방법을 설명합니다.

불명확한 마운트 해제

저널링은 파일 시스템에서 발생하는 메타데이터 변경 사항의 트랜잭션 레코드를 유지 관리합니다.

*시스템 충돌, 전원 장애 또는 기타 불명확한 마운트 해제가 발생하는 경우 **XFS**는 저널(로그라고도 함)을 사용하여 파일 시스템을 복구합니다. 커널은 **XFS** 파일 시스템을 마운트할 때 저널 복구를 수행합니다.*

corruption

이 컨텍스트에서 손상 은 다음과 같이 로 인한 파일 시스템의 오류를 의미합니다.

- *하드웨어 오류*
- *스토리지 펌웨어, 장치 드라이버, 소프트웨어 스택 또는 파일 시스템 자체의 버그*
- *파일 시스템의 일부를 파일 시스템 외부에 의해 덮어쓰는 문제*

***XFS**가 파일 시스템 또는 파일 시스템 메타데이터의 손상을 탐지하면 파일 시스템을 종료하고 시스템 로그에서 문제가 보고될 수 있습니다. **/var** 디렉토리를 호스팅하는 파일 시스템에서 손상이 발생한 경우 재부팅 후 이러한 로그를 사용할 수 없습니다.*

*예 **27.1. XFS** 손상을 보고하는 시스템 로그 항목*

```
# dmesg --notime | tail -15

XFS (loop0): Mounting V5 Filesystem
XFS (loop0): Metadata CRC error detected at xfs_agi_read_verify+0xcb/0xf0 [xfs], xfs_agi block
0x2
XFS (loop0): Unmount and run xfs_repair
XFS (loop0): First 128 bytes of corrupted metadata buffer:
00000000027b3b56: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000005f9abc7a: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000005b0aef35: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000000da9d2ded: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000001e265b07: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000006a40df69: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000000b272907: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000000e484aac5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
XFS (loop0): metadata I/O error in "xfs_trans_read_buf_map" at daddr 0x2 len 1 error 74
XFS (loop0): xfs_imap_lookup: xfs_ialloc_read_agi() returned error -117, agno 0
XFS (loop0): Failed to read root inode 0x80, error 11
```

사용자 공간 유틸리티는 손상된 **XFS** 파일 시스템에 액세스하려고 할 때 일반적으로 **Input/output** 오류 메시지를 보고합니다. 손상된 로그로 **XFS** 파일 시스템을 마운트하면 실패한 마운트와 다음과 같은 오류 메시지가 표시됩니다.

```
mount: /mount-point: mount(2) system call failed: Structure needs cleaning.
```

수동으로 **xfs_repair** 유틸리티를 사용하여 손상을 복구해야 합니다.

추가 리소스

- **xfs_repair(8)** 도움말 페이지.

27.4. XFS_REPAIR로 XFS 파일 시스템 확인

이 절차에서는 **xfs_repair** 유틸리티를 사용하여 **XFS** 파일 시스템의 읽기 전용 검사를 수행합니다. 수동으로 **xfs_repair** 유틸리티를 사용하여 손상을 복구해야 합니다. 다른 파일 시스템 복구 유틸리티와 달리 **xfs_repair** 는 **XFS** 파일 시스템을 완전히 마운트 해제하지 않은 경우에도 부팅 시 실행되지 않습니다. 불명확한 마운트 해제의 경우 **XFS**는 마운트 시 로그를 재생하여 파일 시스템이 일관되게 유지되도록 합니다. **xfs_repair** 는 먼저 다시 마운트하지 않고 더티 로그로 **XFS** 파일 시스템을 복구할 수 없습니다.

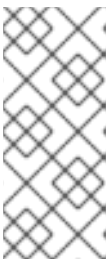
참고

fsck.xfs 바이너리는 **xfsprogs** 패키지에 있지만 부팅 시 **fsck.file** 시스템 바이너리를 찾는 **initscripts** 를 충족하기 위해서만 존재합니다. **fsck.xfs** 는 종료 코드 **0**으로 즉시 종료됩니다.

절차

1. 파일 시스템을 마운트 및 마운트 해제하여 로그를 재생합니다.

```
# mount file-system
# umount file-system
```



참고

구조 정리 오류가 필요한 마운트에 실패하면 로그가 손상되어 복원할 수 없습니다. 예행 실행에서는 결과적으로 더 많은 디스크 손상을 검색하고 보고해야 합니다.

2. **xfs_repair** 유틸리티를 사용하여 파일 시스템을 확인하는 예행행을 수행합니다. 모든 오류가 출력되고 파일 시스템을 수정하지 않고 수행할 작업의 표시가 표시됩니다.

```
# xfs_repair -n block-device
```

3. 파일 시스템을 마운트합니다.

```
# mount file-system
```

추가 리소스

- **xfs_repair(8)** 도움말 페이지.
- **xfs_metadump(8)** 도움말 페이지.

27.5. XFS_REPAIR로 XFS 파일 시스템 복구

이 절차에서는 **xfs_repair** 유틸리티를 사용하여 손상된 **XFS** 파일 시스템을 복구합니다.

절차

1. **xfs_metadump** 유틸리티를 사용하여 진단 또는 테스트 목적으로 복구하기 전에 메타데이터 이미지를 생성합니다. 미리 페어링된 파일 시스템 메타데이터 이미지는 소프트웨어 버그로 인한 손상이 발생하는 경우 조사에 유용할 수 있습니다. 사전 페어링 이미지에 존재하는 손상 패턴은 근본 원인 분석을 지원할 수 있습니다.

   - **xfs_metadump** 디버깅 툴을 사용하여 **XFS** 파일 시스템의 메타데이터를 파일로 복사합니다. 결과적으로 생성되는 메타**dump** 파일을 표준 압축 유틸리티를 사용하여 압축하여 지원할 대규모 메타**dump** 파일을 전송해야 하는 경우 파일 크기를 줄일 수 있습니다.

```
# xfs_metadump block-device metadump-file
```

2. 파일 시스템을 다시 마운트하여 로그를 재생합니다.

```
# mount file-system
# umount file-system
```

3. **xfs_repair** 유틸리티를 사용하여 마운트 해제된 파일 시스템을 복구합니다.

- 마운트가 성공하면 추가 옵션이 필요하지 않습니다.

```
# xfs_repair block-device
```

- **Structure**에서 마운트에 실패한 경우 정리 오류가 필요한 경우 로그가 손상되어 재생될 수 없습니다. **L** 옵션을 사용하여로그를 지우려면 **-L** 옵션(로그 **0ing** 강제)을 사용합니다.

주의

이 명령을 실행하면 충돌 시 모든 메타데이터 업데이트가 손실되어 상당한 파일 시스템 손상 및 데이터 손실이 발생할 수 있습니다. 로그를 재생할 수 없는 경우 마지막 수단으로만 사용해야 합니다.

```
# xfs_repair -L block-device
```

4. 파일 시스템을 마운트합니다.

```
# mount file-system
```

추가 리소스

- **xfs_repair(8)** 도움말 페이지.

27.6. EXT2, EXT3 및 EXT4의 오류 처리 메커니즘

ext2, ext3 및 **ext4** 파일 시스템은 **e2fsck** 유틸리티를 사용하여 파일 시스템 점검 및 복구를 수행합니다. **fsck.ext2,fsck.ext3, fsck.ext4** 파일 이름은 **e2fsck** 유틸리티에 대한 하드 링크입니다. 이러한 바이너리는 부팅 시 자동으로 실행되며 해당 동작은 점검 중인 파일 시스템과 파일 시스템의 상태에 따라 다릅니다.

메타데이터 저널링 파일 시스템이 아닌 **ext2** 및 저널이 없는 **ext4** 파일 시스템에 대한 전체 파일 시스템 검사 및 복구가 호출됩니다.

메타데이터 저널링이 있는 **ext3** 및 **ext4** 파일 시스템의 경우 사용자 공간에서 저널이 재생되고 유틸리티가 종료됩니다. 이는 저널 재생이 충돌 후 일관된 파일 시스템을 보장하므로 기본 작업입니다.

이러한 파일 시스템이 마운트된 동안 메타데이터 불일치가 발생하면 파일 시스템 수퍼 블록에 이 사실을 기록합니다. **e2fsck** 이 파일 시스템이 이러한 오류로 표시된 것을 발견하면 **e2fsck** 은 저널을 재생한 후 전체 검사를 수행합니다(있는 경우).

추가 리소스

- **fsck(8)** 도움말 페이지.
- **e2fsck(8)** 도움말 페이지.

27.7. E2FSCK을 사용하여 EXT2, EXT3 또는 EXT4 파일 시스템 확인

이 절차에서는 **e2fsck** 유틸리티를 사용하여 **ext2, ext3** 또는 **ext4** 파일 시스템을 확인합니다.

절차

1. 파일 시스템을 다시 마운트하여 로그를 재생합니다.

```
# mount file-system
# umount file-system
```

2. 예행 연습을 수행하여 파일 시스템을 확인합니다.

```
# e2fsck -n block-device
```

참고

모든 오류가 출력되고 파일 시스템을 수정하지 않고 수행할 작업의 표시가 표시됩니다. 이후의 일관성 검사 단계는 복구 모드에서 실행 중인 경우 초기 단계에서 수정된 불일치를 발견하면 추가 오류를 출력할 수 있습니다.

추가 리소스

- **e2image(8)** 도움말 페이지.

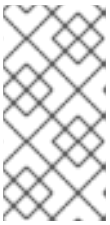
- **e2fsck(8)** 도움말 페이지.

27.8. E2FSCK을 사용하여 EXT2, EXT3 또는 EXT4 파일 시스템 복구

이 절차에서는 **e2fsck** 유틸리티를 사용하여 손상된 **ext2, ext3** 또는 **ext4** 파일 시스템을 복구합니다.

절차

1. 지원 조사를 위해 파일 시스템 이미지를 저장합니다. 미리 페어링된 파일 시스템 메타데이터 이미지는 소프트웨어 버그로 인한 손상이 발생하는 경우 조사에 유용할 수 있습니다. 사전 페어링 이미지에 존재하는 손상 패턴은 근본 원인 분석을 지원할 수 있습니다.



참고

심각하게 손상된 파일 시스템은 메타데이터 이미지 생성에 문제가 발생할 수 있습니다.

- 테스트용으로 이미지를 생성하는 경우 **-r** 옵션을 사용하여 파일 시스템 자체와 동일한 크기의 스파스 파일을 만듭니다. **e2fsck** 은 결과 파일에서 직접 작동할 수 있습니다.

```
# e2image -r block-device image-file
```

- 보관할 이미지를 만들거나 진단을 위해 제공하는 경우 **-Q** 옵션을 사용하여 전송에 적합한 더 컴팩트 파일 형식을 생성합니다.

```
# e2image -Q block-device image-file
```

2. 파일 시스템을 다시 마운트하여 로그를 재생합니다.

```
# mount file-system
# umount file-system
```

3. 파일 시스템을 자동으로 복구합니다. 사용자 개입이 필요한 경우 **e2fsck** 은 출력에서 수정되지 않은 문제를 표시하고 이 상태를 종료 코드에 반영합니다.

```
# e2fsck -p block-device
```

추가 리소스

- ***e2image(8)** 도움말 페이지.*
- ***e2fsck(8)** 도움말 페이지.*

28장. 파일 시스템 마운트

시스템 관리자는 시스템에 파일 시스템을 마운트하여 해당 시스템의 데이터에 액세스할 수 있습니다.

28.1. LINUX 마운트 메커니즘

이 섹션에서는 Linux에서 파일 시스템 마운트에 대한 기본 개념에 대해 설명합니다.

Linux, UNIX 및 유사한 운영 체제에서는 서로 다른 파티션 및 이동식 장치(예: CD, DVD 또는 USB 플래시 드라이브)의 파일 시스템을 디렉토리 트리의 특정 지점(마운트 지점)에 연결한 다음 다시 분리할 수 있습니다. 파일 시스템은 디렉터리에 마운트되지만 디렉터리의 원본 콘텐츠에는 액세스할 수 없습니다.

Linux에서는 파일 시스템이 이미 연결된 디렉터리에 파일 시스템을 마운트할 수 없습니다.

마운트 시 다음을 통해 장치를 식별할 수 있습니다.

- 범용 고유 식별자(UUID): 예: UUID=34795a28-ca6d-4fd8-a347-73671d0c19cb
- 볼륨 레이블(예: LABEL=home)
- 비영구적인 블록 장치의 전체 경로(예: /dev/sda3)

모든 필수 정보 없이 mount 명령을 사용하여 파일 시스템을 마운트할 때 즉 장치 이름, 대상 디렉터리 또는 파일 시스템 유형이 없는 경우 mount 유틸리티는 /etc/fstab 파일의 내용을 읽고 해당 파일 시스템이 나열되는지 확인합니다. /etc/fstab 파일에는 선택한 파일 시스템이 마운트되도록 설정된 장치 이름 목록과 파일 시스템 유형 및 마운트 옵션이 포함되어 있습니다. 따라서 /etc/fstab 에 지정된 파일 시스템을 마운트할 때 다음 명령 구문으로 충분합니다.

- 마운트 지점별 마운트:

```
# mount directory
```

- 블록 장치로 마운트:

```
# mount device
```

추가 리소스

- **mount(8)** 도움말 페이지
- **UUID**와 같은 지속적인 이름 지정 속성을 나열하는 방법.

28.2. 현재 마운트된 파일 시스템 나열

이 절차에서는 현재 마운트된 모든 파일 시스템을 명령줄에 나열하는 방법을 설명합니다.

절차

- 마운트된 모든 파일 시스템을 나열하려면 **findmnt** 유틸리티를 사용합니다.

```
$ findmnt
```

- 나열된 파일 시스템을 특정 파일 시스템 유형으로 제한하려면 **--types** 옵션을 추가합니다.

```
$ findmnt --types fs-type
```

예를 들면 다음과 같습니다.

예 28.1. XFS 파일 시스템만 나열

```
$ findmnt --types xfs

TARGET  SOURCE                                          FSTYPE OPTIONS
/       /dev/mapper/luks-5564ed00-6aac-4406-bfb4-c59bf5de48b5 xfs    rw,relatime
├─/boot /dev/sda1                                       xfs    rw,relatime
└─/home /dev/mapper/luks-9d185660-7537-414d-b727-d92ea036051e xfs    rw,relatime
```

추가 리소스

- **mnt(8)** 도움말 페이지 찾기

28.3. MOUNT를 사용하여 파일 시스템 마운트

다음 절차에서는 **mount** 유틸리티를 사용하여 파일 시스템을 마운트 하는 방법을 설명합니다.

사전 요구 사항

- 선택한 마운트 지점에 이미 마운트된 파일 시스템이 없는지 확인합니다.

```
$ findmnt mount-point
```

절차

1. 특정 파일 시스템을 연결하려면 **mount** 유틸리티를 사용합니다.

```
# mount device mount-point
```

예 28.2. XFS 파일 시스템 마운트

예를 들어 **UUID**로 식별된 로컬 **XFS** 파일 시스템을 마운트하려면 다음을 수행합니다.

```
# mount UUID=ea74bbec-536d-490c-b8d9-5b40bbd7545b /mnt/data
```

2. 마운트 가 파일 시스템 유형을 자동으로 인식할 수 없는 경우 **--types** 옵션을 사용하여 지정합니다.

```
# mount --types type device mount-point
```

예 28.3. NFS 파일 시스템 마운트

예를 들어 원격 **NFS** 파일 시스템을 마운트하려면 다음을 수행합니다.

```
# mount --types nfs4 host:/remote-export /mnt/nfs
```

추가 리소스

- **mount(8)** 도움말 페이지

28.4. 마운트 지점 이동

다음 절차에서는 마운트된 파일 시스템의 마운트 지점을 다른 디렉터리로 변경하는 방법을 설명합니다.

절차

1. 파일 시스템이 마운트된 디렉터리를 변경하려면 다음을 수행합니다.

```
# mount --move old-directory new-directory
```

예 28.4. 홈 파일 시스템 이동

예를 들어 **/mnt/userdirs/** 디렉토리에 마운트된 파일 시스템을 **/home/** 마운트 지점으로 이동하려면 다음을 수행합니다.

```
# mount --move /mnt/userdirs /home
```

2. 파일 시스템이 예상대로 이동되었는지 확인합니다.

```
$ findmnt
$ ls old-directory
$ ls new-directory
```

추가 리소스

- **mount(8)** 도움말 페이지

28.5. UMOUNT로 파일 시스템 마운트 해제

다음 절차에서는 **umount** 유틸리티를 사용하여 파일 시스템을 마운트 해제하는 방법을 설명합니다.

절차

1. 다음 명령 중 하나를 사용하여 파일 시스템을 마운트 해제합니다.

- *마운트 지점의 경우:*

```
# umount mount-point
```

- *모델 번호:*

```
# umount device
```

명령이 다음과 유사한 오류와 함께 실패하면 프로세스에서 리소스를 사용하고 있기 때문에 파일 시스템이 사용 중임을 나타냅니다.

```
umount: /run/media/user/FlashDrive: target is busy.
```

2. *파일 시스템이 사용 중인 경우 **fuser** 유틸리티를 사용하여 액세스하는 프로세스를 결정합니다. 예를 들면 다음과 같습니다.*

```
$ fuser --mount /run/media/user/FlashDrive

/run/media/user/FlashDrive: 18351
```

그런 다음 파일 시스템을 사용하여 프로세스를 종료하고 다시 마운트 해제하십시오.

28.6. 일반적인 마운트 옵션

다음 표에는 마운트 유틸리티의 가장 일반적인 옵션이 나열되어 있습니다. 다음 구문을 사용하여 이러한 마운트 옵션을 적용할 수 있습니다.

```
# mount --options option1,option2,option3 device mount-point
```

*표 **28.1.** 일반적인 마운트 옵션*

옵션	설명
async	파일 시스템에서 비동기 입력 및 출력 작업을 활성화합니다.
auto	**mount -a** 명령을 사용하여 파일 시스템을 자동으로 마운트할 수 있습니다.
defaults	**async,auto,dev,exec,nouser,rw,suid** 옵션의 별칭을 제공합니다.

옵션	설명
exec	특정 파일 시스템에서 바이너리 파일을 실행할 수 있습니다.
loop	이미지를 루프 장치로 마운트합니다.
noauto	기본 동작은 **mount -a** 명령을 사용하여 파일 시스템의 자동 마운트를 비활성화합니다.
noexec	특정 파일 시스템에서 바이너리 파일의 실행을 허용하지 않습니다.
nouser	파일 시스템을 마운트 및 마운트 해제하기 위해 root 이외의 일반 사용자(즉, root 이외의)를 허용하지 않습니다.
reinstall	이미 마운트된 경우 파일 시스템을 다시 마운트합니다.
ro	읽기 전용으로 파일 시스템을 마운트합니다.
rw	읽기 및 쓰기를 위해 파일 시스템을 마운트합니다.
user	일반 사용자(즉, root가 아닌)가 파일 시스템을 마운트 및 마운트 해제할 수 있습니다.

29장. 여러 마운트 지점에 마운트 공유

시스템 관리자는 마운트 지점을 복제하여 여러 디렉터리에서 파일 시스템에 액세스할 수 있도록 할 수 있습니다.

29.1. 공유 마운트 유형

사용할 수 있는 여러 유형의 공유 마운트가 있습니다. 차이점은 공유 마운트 지점 중 하나에 다른 파일 시스템을 마운트할 때 발생하는 작업입니다. 공유 마운트는 공유 하위 트리 기능을 사용하여 구현됩니다.

다음 마운트 유형을 사용할 수 있습니다.

private

이 유형은 **propagation** 이벤트를 수신하거나 전달하지 않습니다.

다른 파일 시스템을 중복 또는 원래 마운트 지점에 마운트하면 다른 파일 시스템에 반영되지 않습니다.

shared

이 유형은 지정된 마운트 지점의 정확한 복제본을 생성합니다.

마운트 지점을 공유 마운트로 표시하면 원래 마운트 지점 내의 모든 마운트가 반영되며 그 반대의 경우도 마찬가지입니다.

루트 파일 시스템의 기본 마운트 유형입니다.

slave

이 유형은 지정된 마운트 지점의 제한된 중복을 생성합니다.

마운트 지점을 슬레이브 마운트로 표시하면 원래 마운트 지점 내의 모든 마운트가 반영되지만 슬레이브 마운트 내의 마운트는 원래에 반영되지 않습니다.

unbindable

이 유형은 지정된 마운트 지점이 중복되지 않도록 합니다.

추가 리소스

- Linux Weekly news에 대한 Shared subtrees 문서.

29.2. 개인 마운트 지점 중복 생성

이 절차에서는 마운트 지점을 개인 마운트로 중복합니다. 나중에 중복 아래에 마운트하거나 원래 마운트 지점이 다른 마운트 지점에 반영되지 않은 파일 시스템은 다음과 같습니다.

절차

1. 원래 마운트 지점에 가상 파일 시스템(VFS) 노드를 생성합니다.

```
# mount --bind original-dir original-dir
```

2. 원래 마운트 지점을 비공개로 표시합니다.

```
# mount --make-private original-dir
```

또는 선택한 마운트 지점의 마운트 유형과 해당 마운트 지점의 모든 마운트 유형을 변경하려면 **--make-private** 대신 **--make-rprivate** 옵션을 사용합니다.

3. 중복을 생성합니다.

```
# mount --bind original-dir duplicate-dir
```

예 29.1. 개인 마운트 지점으로 /media를 /mnt로 중복

1. /media 디렉토리에서 NetNamespace 노드를 생성합니다.

```
# mount --bind /media /media
```

2. /media 디렉터리를 비공개로 표시합니다.

```
# mount --make-private /media
```

3. */mnt 에 중복을 생성합니다.*

```
# mount --bind /media /mnt
```

4. *이제 /media 및 /mnt 공유 내용이 있지만 /media 내의 마운트 중 어느 것도 /mnt 에 표시되지 않는지 확인할 수 있습니다. 예를 들어 CD-ROM 드라이브에 비어 있지 않은 미디어와 /media/cdrom/ 디렉터리가 있는 경우 다음을 사용합니다.*

```
# mount /dev/cdrom /media/cdrom
# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
# ls /mnt/cdrom
#
```

5. */mnt 디렉토리에 마운트된 파일 시스템이 /media 에도 반영되지 않았는지 확인할 수 있습니다. 예를 들어 /dev/sdc1 장치를 사용하는 비어 있지 않은 USB플러그 드라이브가 연결되어 있고 /mnt/flashdisk/ 디렉터리가 있는 경우 다음을 사용합니다.*

```
# mount /dev/sdc1 /mnt/flashdisk
# ls /media/flashdisk
# ls /mnt/flashdisk
en-US  publican.cfg
```

추가 리소스

- *mount(8) 도움말 페이지*

29.3. 공유 마운트 지점 중복 생성

이 절차에서는 마운트 지점을 공유 마운트로 중복합니다. 나중에 원래 디렉토리 아래에 마운트되거나 중복된 파일 시스템은 항상 다른 디렉터리에 반영됩니다.

절차

1. *원래 마운트 지점에 가상 파일 시스템(VFS) 노드를 생성합니다.*

```
# mount --bind original-dir original-dir
```

2. 원래 마운트 지점을 공유로 표시합니다.

```
# mount --make-shared original-dir
```

또는 선택한 마운트 지점의 마운트 유형과 해당 마운트 지점의 모든 마운트 유형을 변경하려면 **--make-shared** 대신 **--make-rshared** 옵션을 사용합니다.

3. 중복을 생성합니다.

```
# mount --bind original-dir duplicate-dir
```

예 29.2. 공유 마운트 지점으로 **/media**를 **/mnt**로 중복

/media 및 **/mnt** 디렉터리를 동일한 콘텐츠를 공유하도록 하려면 다음을 수행합니다.

1. **/media** 디렉토리에서 **NetNamespace** 노드를 생성합니다.

```
# mount --bind /media /media
```

2. **/media** 디렉터리를 **shared**로 표시합니다.

```
# mount --make-shared /media
```

3. **/mnt** 에 중복을 생성합니다.

```
# mount --bind /media /mnt
```

4. 이제 **/media** 내에 마운트가 **/mnt** 에도 표시되는지 확인할 수 있습니다. 예를 들어 **CD-ROM** 드라이브에 비어 있지 않은 미디어와 **/media/cdrom/** 디렉터리가 있는 경우 다음을 사용합니다.

```
# mount /dev/cdrom /media/cdrom
# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
# ls /mnt/cdrom
EFI  GPL  isolinux  LiveOS
```

5.

마찬가지로 **/mnt** 디렉터리에 마운트된 파일 시스템이 **/media** 에 반영되었는지 확인할 수 있습니다. 예를 들어 **/dev/sdc1** 장치를 사용하는 비어 있지 않은 **USB**플러그 드라이브가 연결되어 있고 **/mnt/flashdisk/** 디렉터리가 있는 경우 다음을 사용합니다.

```
# mount /dev/sdc1 /mnt/flashdisk
# ls /media/flashdisk
en-US  publican.cfg
# ls /mnt/flashdisk
en-US  publican.cfg
```

추가 리소스

- **mount(8)** 도움말 페이지

29.4. 슬레이브 마운트 지점 중복 생성

이 절차에서는 마운트 지점을 슬레이브 마운트 유형으로 중복합니다. 나중에 원래 마운트 지점 아래에 마운트되는 파일 시스템은 중복에 반영되지만 다른 방법은 반영되지 않습니다.

절차

1. 원래 마운트 지점에 가상 파일 시스템**(VFS)** 노드를 생성합니다.

```
# mount --bind original-dir original-dir
```

2. 원래 마운트 지점을 공유로 표시합니다.

```
# mount --make-shared original-dir
```

또는 선택한 마운트 지점의 마운트 유형과 해당 마운트 지점의 모든 마운트 유형을 변경하려면 **--make-shared** 대신 **--make-rshared** 옵션을 사용합니다.

3. 중복을 생성하고 이를 슬레이브 유형으로 표시합니다.

```
# mount --bind original-dir duplicate-dir
# mount --make-slave duplicate-dir
```

예 **29.3.** 슬레이브 마운트 지점으로 **/media**를 **/mnt**에 중복

*이 예에서는 **/media** 디렉터리의 콘텐츠를 **/mnt** 에 표시하지만 **/mnt** 디렉토리에 마운트하지 않고 **/media** 에 반영하는 방법을 보여줍니다.*

1. *__/media__ 디렉토리에서 __NetNamespace__ 노드를 생성합니다.*

```
# mount --bind /media /media
```

2. */__media__ 디렉터리를 __shared__로 표시합니다.*

```
# mount --make-shared /media
```

3. */__mnt__ 에 중복을 생성하고 이를 슬레이브 로 표시합니다.*

```
# mount --bind /media /mnt
# mount --make-slave /mnt
```

4. */__media__ 내의 마운트가 /__mnt__ 에도 표시되는지 확인합니다. 예를 들어 __CD-ROM__ 드라이브에 비어 있지 않은 미디어와 /__media__/__cdrom__/ 디렉터리가 있는 경우 다음을 사용합니다.*

```
# mount /dev/cdrom /media/cdrom
# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
# ls /mnt/cdrom
EFI  GPL  isolinux  LiveOS
```

5. *또한 /__mnt__ 디렉터리에 마운트된 파일 시스템이 /__media__ 에 반영되지 않았는지 확인합니다. 예를 들어 /__dev__/__sdc1__ 장치를 사용하는 비어 있지 않은 __USB__플러그 드라이브가 연결되어 있고 /__mnt__/__flashdisk__/ 디렉터리가 있는 경우 다음을 사용합니다.*

```
# mount /dev/sdc1 /mnt/flashdisk
# ls /media/flashdisk
# ls /mnt/flashdisk
en-US  publican.cfg
```

추가 리소스

- *__mount(8)__ 도움말 페이지*

29.5. 마운트 지점이 중복되지 않도록 방지

이 절차에서는 마운트 지점을 다른 마운트 지점에서 복제할 수 없도록 마운트 지점을 바인딩할 수 없음으로 표시합니다.

절차

- 마운트 지점의 유형을 바인딩을 해제할 수 없는 마운트로 변경하려면 다음을 사용합니다.

```
# mount --bind mount-point mount-point
# mount --make-unbindable mount-point
```

또는 선택한 마운트 지점 및 그 아래의 모든 마운트 지점의 마운트 유형을 변경하려면 **--make-unbindable** 대신 **--make-runbindable** 옵션을 사용합니다.

이 마운트를 중복하려고 하면 다음 오류와 함께 후속 시도가 실패합니다.

```
# mount --bind mount-point duplicate-dir

mount: wrong fs type, bad option, bad superblock on mount-point,
missing codepage or helper program, or other error
In some cases useful info is found in syslog - try
dmesg | tail  or so
```

예 29.4. /media가 중복되지 않도록 방지

- **/media** 디렉터리가 공유되지 않도록 하려면 다음을 사용합니다.

```
# mount --bind /media /media
# mount --make-unbindable /media
```

추가 리소스

- **mount(8)** 도움말 페이지

30장. 파일 시스템 영구적으로 마운트

시스템 관리자는 영구적으로 파일 시스템을 마운트하여 복구할 수 없는 스토리지를 구성할 수 있습니다.

30.1. /ETC/FSTAB 파일

이 섹션에서는 파일 시스템의 영구 마운트 지점을 제어하는 **/etc/fstab** 구성 파일에 대해 설명합니다. **/etc/fstab** 를 사용하여 파일 시스템을 영구적으로 마운트하는 것이 좋습니다.

/etc/fstab 파일의 각 행은 파일 시스템의 마운트 지점을 정의합니다. 여기에는 공백으로 구분된 **6**개의 필드가 있습니다.

1. 영구 속성 또는 경로별로 **/dev** 디렉터리인 블록 장치입니다.
2. 장치가 마운트될 디렉터리입니다.
3. 장치의 파일 시스템입니다.
4. 파일 시스템의 마운트 옵션입니다. 옵션 기본값은 파티션이 부팅 시 기본 옵션으로 마운트됨을 의미합니다. 이 섹션에서는 **x-systemd.** 옵션 형식의 **systemd** 마운트 장치옵션 도 인식합니다.
5. 덤프 유틸리티의 백업 옵션입니다.
6. **fsck** 유틸리티의 순서를 확인합니다.

참고

파일 시스템 백업에 사용되는 **dump** 유틸리티는 **RHEL 9**에서 제거되었으며 **EPEL 9** 리포지토리에서 사용할 수 있습니다.

예 30.1. /etc/fstab의 /boot 파일 시스템

블록 장치	마운트 지점	파일 시스템	옵션	Backup	검사
UUID=ea74bbec-536d-490c-b8d9-5b40bbd7545b	**/boot**	**xfs**	**defaults**	**0**	**0**

systemd 서비스는 /etc/fstab 의 항목에서 마운트 단위를 자동으로 생성합니다.

추가 리소스

- *fstab(5) 도움말 페이지*
- *systemd.mount(5) 도움말 페이지*

30.2. /ETC/FSTAB에 파일 시스템 추가

다음 절차에서는 /etc/fstab 구성 파일에서 파일 시스템의 영구 마운트 지점을 구성하는 방법을 설명합니다.

절차

1. *파일 시스템의 UUID 특성을 찾습니다.*

```
$ lsblk --fs storage-device
```

예를 들면 다음과 같습니다.

예 30.2. 파티션의 UUID 보기

```
$ lsblk --fs /dev/sda1

NAME FSTYPE LABEL UUID                                 MOUNTPOINT
sda1 xfs    Boot  ea74bbec-536d-490c-b8d9-5b40bbd7545b /boot
```

2. *마운트 지점 디렉터리가 없으면 생성합니다.*

```
# mkdir --parents mount-point
```

3. **root**로 **/etc/fstab** 파일을 편집하고 **UUID**로 식별되는 파일 시스템의 행을 추가합니다.

   예를 들면 다음과 같습니다.

   예 30.3. /etc/fstab의 /boot 마운트 지점

   ```
   UUID=ea74bbec-536d-490c-b8d9-5b40bbd7545b /boot xfs defaults 0 0
   ```

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

   ```
   # systemctl daemon-reload
   ```

5. 파일 시스템을 마운트하여 구성이 작동하는지 확인합니다.

   ```
   # mount mount-point
   ```

추가 리소스

- 영구 이름 지정 특성 개요.

31장. RHEL 시스템 역할을 사용하여 파일 시스템 영구적으로 마운트

이 섹션에서는 스토리지 역할을 사용하여 파일 시스템을 영구적으로 마운트하는 방법을 설명합니다.

사전 요구 사항

- 스토리지 역할을 사용하는 **Ansible** 플레이북이 있습니다.

31.1. 파일 시스템을 영구적으로 마운트하는 ANSIBLE 플레이북의 예

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 **storage** 역할을 적용하여 **XFS** 파일 시스템을 즉시 지속적으로 마운트합니다.

예 31.1. /dev/sdb에 파일 시스템을 /mnt/data에 마운트하는 플레이북

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- 이 **Playbook**은 **/etc/fstab** 파일에 파일 시스템을 추가하고 파일 시스템을 즉시 마운트합니다.
- **/dev/sdb** 장치 또는 마운트 지점 디렉터리의 파일 시스템이 존재하지 않는 경우 플레이북에서 해당 시스템을 생성합니다.

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일.

32장. 필요에 따라 파일 시스템 마운트

시스템 관리자는 NFS와 같은 파일 시스템을 구성하여 필요에 따라 자동으로 마운트할 수 있습니다.

32.1. EGRESSIP 서비스

이 섹션에서는 필요에 따라 파일 시스템을 마운트하는 데 사용되는 EgressIP 서비스의 이점과 기본 개념에 대해 설명합니다.

/etc/fstab 구성을 사용한 영구 마운트의 한 가지 단점은 사용자가 종종 마운트된 파일 시스템에 액세스하는 방법에 관계없이 마운트된 파일 시스템을 유지하기 위해 리소스를 전용으로 지정해야 한다는 것입니다. 예를 들어 시스템은 한 번에 여러 시스템에 NFS 마운트를 유지 관리하는 경우 시스템 성능에 영향을 미칠 수 있습니다.

/etc/fstab 에 대한 대안은 커널 기반 EgressIP 서비스를 사용하는 것입니다. 다음 구성 요소로 구성됩니다.

- 파일 시스템을 구현하는 커널 모듈
- 다른 모든 기능을 수행하는 사용자 공간 서비스입니다.

E gressIP 서비스는 파일 시스템을 자동으로 마운트 및 마운트 해제하여 시스템 리소스를 저장할 수 있습니다. NFS, AFS, SMBFS, CIFS 및 로컬 파일 시스템과 같은 파일 시스템을 마운트하는 데 사용할 수 있습니다.

추가 리소스

- E gressIP(8) 도움말 페이지.

32.2. EGRESSIP 구성 파일

이 섹션에서는 EgressIP 서비스에서 사용하는 구성 파일의 사용법 및 구문에 대해 설명합니다.

마스터 맵 파일

E gressIP 서비스는 /etc/auto.master (마스터 맵)를 기본 구성 파일로 사용합니다. 이렇게 하면 이름

서비스 스위치(NSS) 메커니즘과 함께 /etc/ autofs.conf 구성 파일의 sssd 설정을 사용하여 지원되는 다른 네트워크 소스와 이름을 사용하도록 변경할 수 있습니다.

모든 온 디맨드 마운트 지점은 마스터 맵에서 구성해야 합니다. 마운트 지점, 호스트 이름, 내보낸 디렉토리 및 옵션은 각 호스트에 대해 수동으로 구성하지 않고 파일 집합(또는 기타 지원되는 네트워크 소스)에 지정할 수 있습니다.

마스터 맵 파일은 EgressIP 에서 제어하는 마운트 지점과 자동 마운트 맵이라는 해당 구성 파일 또는 네트워크 소스를 나열합니다. 마스터 맵의 형식은 다음과 같습니다.

```
mount-point  map-name  options
```

이 형식으로 사용되는 변수는 다음과 같습니다.

Mount-point

autofs 마운트 지점(예: /mnt/data)입니다.

map-file

마운트 지점 목록과 해당 마운트 지점을 마운트해야 하는 파일 시스템 위치가 포함된 맵 소스 파일입니다.

옵션

제공되는 경우 이러한 항목은 지정된 옵션이 없는 경우 지정된 맵의 모든 항목에 적용됩니다.

예 32.1. /etc/auto.master 파일

다음은 /etc/auto.master 파일의 샘플 행입니다.

```
/mnt/data  /etc/auto.data
```

파일 매핑

맵 파일은 개별 온 디맨드 마운트 지점의 속성을 구성합니다.

자동 마운터는 디렉터리가 없는 경우 해당 디렉터리를 생성합니다. 자동 마운터가 시작되기 전에 디렉터리가 있는 경우 자동 마운터는 종료 시 해당 디렉토리를 제거하지 않습니다. 시간 초과를 지정하면 시간

초과 기간에 디렉터리에 액세스하지 않으면 디렉터리가 자동으로 마운트 해제됩니다.

일반 맵 형식은 마스터 맵과 유사합니다. 그러나 **options** 필드는 마스터 맵에서와 같이 항목의 끝에 있는 대신 마운트 지점과 위치 사이에 나타납니다.

```
mount-point  options  location
```

이 형식으로 사용되는 변수는 다음과 같습니다.

Mount-point

resulting 마운트 지점 을 나타냅니다. 간접 마운트의 단일 디렉토리 이름 또는 직접 마운트를 위한 마운트 지점의 전체 경로일 수 있습니다. 각 직접 및 간접 맵 항목(**mount-point**)을 뒤에 공백으로 구분된 오프셋 디렉토리 목록(/으로 시작하는 하위 디렉토리 이름)을 사용하여 다중 마운트 항목으로 만들 수 있습니다.

옵션

제공된 경우 이러한 옵션은 마스터 맵 항목 옵션(있는 경우)에 추가되거나 구성 항목 **append_options** 가 **no** 로 설정된 경우 마스터 맵 옵션 대신 사용됩니다.

위치

이는 로컬 파일 시스템 경로와 같은 파일 시스템 위치(**Sun map** 형식 이스케이프 문자 **:** /로 시작하는 맵 이름), **NFS** 파일 시스템 또는 기타 유효한 파일 시스템 위치와 같은 파일 시스템 위치를 나타냅니다.

예 32.2. 맵 파일

다음은 맵 파일의 샘플입니다(예: **/etc/auto.misc**):

```
payroll  -fstype=nfs4  personnel:/exports/payroll
sales    -fstype=xfs   :/dev/hda4
```

맵 파일의 첫 번째 열은 **EgressIP** 마운트 지점을 나타냅니다. **sales** 및 **payroll from the server called staff** . 두 번째 열은 **EgressIP** 마운트 옵션을 나타냅니다. 세 번째 열은 마운트의 소스를 나타냅니다.

지정된 설정에 따라 **metadata** 마운트 지점 은 **/home/ paidroll** 및 **/home/ business** 임을 의미합니다. **fs type=** 옵션은 종종 생략되며, 시스템 기본값이 **NFS** 마운트의 경우 **NFSv4**에 대한 마운트를 포함하여 파일 시스템이 **NFS**인 경우에는 필요하지 않습니다.

지정된 구성을 사용하여 프로세스에서 /home/ paidroll/2006/July.sxc 와 같은 unmounted 디렉터리에 대한 액세스 권한이 필요한 경우 EgressIP 서비스는 디렉터리를 자동으로 마운트합니다.

amd map 형식

E gressIP 서비스는 amd 형식으로 된 맵 구성을 인식합니다. 이는 Red Hat Enterprise Linux에서 제거된 am-utils 서비스에 작성된 기존 자동 마운터 구성을 재사용하려는 경우에 유용합니다.

그러나 Red Hat은 이전 섹션에 설명된 간단한 result ing 형식을 사용하는 것이 좋습니다.

추가 리소스

- EgressIP(5) 도움말 페이지
- EgressIP.conf(5) 도움말 페이지
- auto.master(5) 도움말 페이지
- /usr/share/doc/autofs/README.amd-maps file

32.3. EGRESSIP 마운트 지점 구성

다음 절차에서는 EgressIP 서비스를 사용하여 주문형 마운트 지점을 구성하는 방법을 설명합니다.

사전 요구 사항

- EgressIP 패키지를 설치합니다.

```
# dnf install autofs
```

- EgressIP 서비스를 시작하고 활성화합니다.

```
# systemctl enable --now autofs
```

절차

1. /etc/auto.식별자에있는 온디맨드 마운트 지점의 맵 파일을 만듭니다. ID 를 마운트 지점을 식별하는 이름으로 바꿉니다.

2. map 파일에서 The autofs 설정 파일에 설명된 대로 마운트 지점, 옵션, 위치 필드를 입력합니다.

3. The autofs 설정 파일에 설명된 대로 마스터 맵 파일에 맵 파일을 등록합니다.

4. 서비스에서 구성을 다시 읽도록 허용하여 새로 구성된 autofs 마운트를 관리할 수 있습니다.

```
# systemctl reload autofs.service
```

5. 온 디맨드 디렉터리의 콘텐츠에 액세스해 보십시오.

```
# ls automounted-directory
```

32.4. EGRESSIP 서비스로 NFS 서버 사용자 홈 디렉토리 자동 마운트

다음 절차에서는 사용자 홈 디렉터리를 자동으로 마운트하도록 EgressIP 서비스를 구성하는 방법을 설명합니다.

사전 요구 사항

- E gressIP 패키지가 설치되어 있습니다.
- E gressIP 서비스가 활성화되어 실행되고 있습니다.

절차

1. 사용자 홈 디렉토리를 마운트해야 하는 서버에서 /etc/auto.master 파일을 편집하여 맵 파일의 마운트 지점과 위치를 지정합니다. 이렇게 하려면 /etc/auto.master 파일에 다음 행을 추가합니다.

```
/home /etc/auto.home
```

2. 사용자 홈 디렉터리를 마운트해야 하는 서버에서 **/etc/auto.home** 이라는 이름으로 맵 파일을 만들고 다음 매개 변수를 사용하여 파일을 편집합니다.

```
* -fstype=nfs,rw,sync host.example.com:/home/&
```

기본적으로 **nfs** 이므로 **fstype** 매개변수를 건너뛸 수 있습니다. 자세한 내용은 **EgressIP (5)** 매뉴얼 페이지를 참조하십시오.

3. **EgressIP** 서비스를 다시 로드합니다.

```
# systemctl reload autofs
```

32.5. SSSD 사이트 구성 파일 덮어쓰기 또는 보강

클라이언트 시스템의 특정 마운트 지점에 대한 사이트 기본값을 재정의하는 것이 유용한 경우가 있습니다.

예 32.3. 초기 조건

예를 들어 다음 조건을 고려하십시오.

- 자동 마운터 맵은 **NIS**에 저장되고 **/etc/nsswitch.conf** 파일에는 다음 지시문이 있습니다.

```
automount:    files nis
```

- **auto.master** 파일에는 다음이 포함됩니다.

```
+auto.master
```

- **NIS auto.master** 맵 파일은 다음과 같습니다.

```
/home auto.home
```

- **NIS auto.home** 맵은 다음과 같습니다.

```
beth    fileserver.example.com:/export/home/beth
joe     fileserver.example.com:/export/home/joe
*       fileserver.example.com:/export/home/&
```

- **autofs** 설정 옵션 **BROWSE_MODE** 가 **yes** 로 설정되어 있습니다.

```
BROWSE_MODE="yes"
```

- 파일 맵 **/etc/auto.home** 이 존재하지 않습니다.

절차

이 섹션에서는 다른 서버에서 홈 디렉토리를 마운트하고 선택한 항목만으로 **auto.home** 을 늘리는 예제에 대해 설명합니다.

예 32.4. 다른 서버의 홈 디렉토리 마운트

이전 조건을 고려할 때 클라이언트 시스템이 **NIS** 맵 **auto.home** 을 재정의하고 다른 서버의 홈 디렉터리를 마운트해야 한다고 가정하겠습니다.

- 이 경우 클라이언트는 다음 **/etc/auto.master** 맵을 사용해야 합니다.

```
/home /etc/auto.home
+auto.master
```

- **/etc/auto.home** 맵에는 항목이 포함되어 있습니다.

```
*    host.example.com:/export/home/&
```

자동 마운터는 마운트 지점의 첫 번째 발생만 처리하므로 **/home** 디렉터리에는 **NIS auto.home** 맵 대신 **/etc/auto.home** 의 콘텐츠가 포함되어 있습니다.

예 32.5. 선택한 항목만 사용하여 **auto.home** 추가

또는 사이트 전체 **auto.home** 맵을 몇 개의 항목으로 보강하려면 다음을 수행합니다.

1. */etc/auto.home 파일 맵을 만들고 여기에 새 항목을 배치합니다. 결국 NIS auto.home 맵을 포함합니다. 그런 다음 /etc/auto.home 파일 맵은 다음과 유사합니다.*

```
mydir someserver:/export/mydir
+auto.home
```

2. *이러한 NIS auto.home 맵 조건을 통해 /home 디렉토리 출력의 내용을 나열하십시오.*

```
$ ls /home

beth joe mydir
```

이 마지막 예는 sssd 가 읽는 것과 동일한 이름의 파일 맵 내용을 포함하지 않기 때문에 예상대로 작동합니다. 이와 같이 E gressIP 은 nsswitch 구성의 다음 맵 소스로 이동합니다.

32.6. LDAP를 사용하여 자동 마운터 맵 저장

이 절차에서는 E gressIP 맵 파일에 있는 것이 아니라 LDAP 구성에 자동 마운터 맵을 저장하도록 E gressIP를 구성합니다.

사전 요구 사항

- *LDAP 클라이언트 라이브러리는 LDAP에서 mount map을 검색하도록 구성된 모든 시스템에 설치되어 있어야 합니다. Red Hat Enterprise Linux에서 openldap 패키지는 EgressIP 패키지의 종속성으로 자동으로 설치되어야 합니다.*

절차

1. *LDAP 액세스를 구성하려면 /etc/openldap/ldap.conf 파일을 수정합니다. BASE,URI, 스키마 옵션이 사이트에 적절하게 설정되어 있는지 확인합니다.*

2. *LDAP에 자동 마운트 맵을 저장하기 위한 가장 최근의 스키마는 rfc2307bis 초안에 설명되어 있습니다. 이 스키마를 사용하려면 스키마 정의에서 주석 문자를 제거하여 /etc/autofs.conf 구성 파일에서 설정합니다. 예를 들면 다음과 같습니다.*

예 32.6. EgressIP 구성 설정

```
DEFAULT_MAP_OBJECT_CLASS="automountMap"
DEFAULT_ENTRY_OBJECT_CLASS="automount"
```

```
DEFAULT_MAP_ATTRIBUTE="automountMapName"
DEFAULT_ENTRY_ATTRIBUTE="automountKey"
DEFAULT_VALUE_ATTRIBUTE="automountInformation"
```

3. 구성에 다른 모든 스키마 항목이 주석 처리되었는지 확인합니다. **rfc2307bis** 스키마의 **automountKey** 속성은 **rfc2307** 스키마의 **cn** 속성을 대체합니다 . 다음은 **LDAP Data Interchange Format(LDIF)** 구성의 예입니다.

예 32.7. LDIF 설정

```
# auto.master, example.com
dn: automountMapName=auto.master,dc=example,dc=com
objectClass: top
objectClass: automountMap
automountMapName: auto.master

# /home, auto.master, example.com
dn: automountMapName=auto.master,dc=example,dc=com
objectClass: automount
automountKey: /home
automountInformation: auto.home

# auto.home, example.com
dn: automountMapName=auto.home,dc=example,dc=com
objectClass: automountMap
automountMapName: auto.home

# foo, auto.home, example.com
dn: automountKey=foo,automountMapName=auto.home,dc=example,dc=com
objectClass: automount
automountKey: foo
automountInformation: filer.example.com:/export/foo

# /, auto.home, example.com
dn: automountKey=/,automountMapName=auto.home,dc=example,dc=com
objectClass: automount
automountKey: /
automountInformation: filer.example.com:/export/&
```

추가 리소스

- *rfc2307bis 초안*

32.7. SYSTEMD.AUTOMOUNT를 사용하여 필요에 따라 /ETC/FSTAB에 파일 시스템 마운트

다음 절차에서는 마운트 지점이 **/etc/fstab** 에 정의된 경우 자동 마운트 **systemd** 장치를 사용하여 필요에 따라 파일 시스템을 마운트하는 방법을 보여줍니다. 각 마운트에 대해 자동 마운트 장치를 추가하고 활

성화해야 합니다.

절차

1. *Chapter 30에 설명된 대로 원하는 fstab 항목을 추가합니다. 파일 시스템 영구적으로 마운트. 예를 들면 다음과 같습니다.*

```
/dev/disk/by-id/da875760-edb9-4b82-99dc-5f4b1ff2e5f4  /mount/point  xfs  defaults  0 0
```

2. *이전 단계에서 만든 항목의 options 필드에 x-systemd.automount 를 추가합니다.*

3. *시스템이 새 구성을 등록하도록 새로 생성된 장치를 로드합니다.*

```
# systemctl daemon-reload
```

4. *자동 마운트 장치를 시작합니다.*

```
# systemctl start mount-point.automount
```

검증

1. *mount-point.automount 가 실행 중인지 확인합니다.*

```
# systemctl status mount-point.automount
```

2. *자동 마운트된 디렉토리에 필요한 콘텐츠가 있는지 확인합니다.*

```
# ls /mount/point
```

추가 리소스

- *systemd.automount(5) 도움말 페이지.*
- *systemd.mount(5) 도움말 페이지.*

- ***systemd 소개*.**

32.8. SYSTEMD.AUTOMOUNT를 사용하여 마운트 단위로 필요에 따라 파일 시스템 마운트

다음 절차에서는 마운트 지점을 마운트할 때 마운트 단위를 사용하여 필요에 따라 파일 시스템을 마운트하는 방법을 보여줍니다. 각 마운트에 대해 자동 마운트 장치를 추가하고 활성화해야 합니다.

절차

1. *마운트 장치를 생성합니다. 예를 들면 다음과 같습니다.*

```
mount-point.mount
[Mount]
What=/dev/disk/by-uuid/f5755511-a714-44c1-a123-cfde0e4ac688
Where=/mount/point
Type=xfs
```

2. *마운트 단위와 이름이 동일하지만 확장자는 **.automount** 인 유닛 파일을 만듭니다.*

3. *파일을 열고 **[자동 마운트]** 섹션을 만듭니다. **Where=** 옵션을 마운트 경로로 설정합니다.*

```
[Automount]
Where=/mount/point
[Install]
WantedBy=multi-user.target
```

4. *시스템이 새 구성을 등록하도록 새로 생성된 장치를 로드합니다.*

```
# systemctl daemon-reload
```

5. *대신 자동 마운트 장치를 활성화하고 시작합니다.*

```
# systemctl enable --now mount-point.automount
```

검증

1. ***mount-point.automount** 가 실행 중인지 확인합니다.*

```
# systemctl status mount-point.automount
```

2. 자동 마운트된 디렉토리에 필요한 콘텐츠가 있는지 확인합니다.

```
# ls /mount/point
```

추가 리소스

- **systemd.automount(5)** 도움말 페이지.
- **systemd.mount(5)** 도움말 페이지.
- **systemd** 소개.

33장. IDM의 SSSD 구성 요소를 사용하여 EGRESSIP 맵 캐시

SSSD(System Security Services Daemon)는 원격 서비스 디렉터리 및 인증 메커니즘에 액세스하기 위한 시스템 서비스입니다. 데이터 캐싱은 네트워크 연결 속도가 느린 경우 유용합니다. EgressIP 맵을 캐시하도록 SSSD 서비스를 구성하려면 이 섹션의 다음 절차를 따르십시오.

33.1. IDM 서버를 LDAP 서버로 사용하도록 수동 설정

다음 절차에서는 IdM 서버를 LDAP 서버로 사용하도록 EgressIP을 구성하는 방법을 설명합니다.

절차

1. /etc/autofs.conf 파일을 편집하여 EgressIP에서 검색하는 스키마 속성 을 지정합니다.

```
#
# Other common LDAP naming
#
map_object_class = "automountMap"
entry_object_class = "automount"
map_attribute = "automountMapName"
entry_attribute = "automountKey"
value_attribute = "automountInformation"
```

참고

사용자는 /etc/autofs.conf 파일의 하위 및 상위 케이스 모두에서 속성을 작성할 수 있습니다.

2. 필요한 경우 LDAP 구성을 지정합니다. 이 작업을 수행하는 방법은 두 가지가 있습니다. 가장 간단한 방법은 자동 마운트 서비스에서 LDAP 서버와 위치를 자체적으로 검색할 수 있도록 하는 것입니다.

```
ldap_uri = "ldap:///dc=example,dc=com"
```

이 옵션을 사용하려면 DNS에서 검색 가능한 서버에 대한 SRV 레코드를 포함해야 합니다.

또는 사용할 LDAP 서버와 LDAP 검색의 기본 DN을 명시적으로 설정합니다.

```
ldap_uri = "ldap://ipa.example.com"
search_base = "cn=location,cn=automount,dc=example,dc=com"
```

3. EgressIP이 IdM LDAP 서버를 사용하여 클라이언트 인증을 허용하도록 /etc/autofs_ldap_auth.conf 파일을 편집합니다.

- auth required 를 yes로 변경합니다.
- IdM LDAP 서버의 Kerberos 호스트 주체인 host/fqdn@REALM 으로 주체를 설정합니다. 주체 이름은 GSS 클라이언트 인증의 일부로 IdM 디렉터리에 연결하는 데 사용됩니다.

```
<autofs_ldap_sasl_conf
    usetls="no"
    tlsrequired="no"
    authrequired="yes"
    authtype="GSSAPI"
    clientprinc="host/server.example.com@EXAMPLE.COM"
    />
```

호스트 주체에 대한 자세한 내용은 IdM에서 표준 DNS 호스트 이름 사용을 참조하십시오.

필요한 경우 klist -k 를 실행하여 정확한 호스트 주체 정보를 가져옵니다.

33.2. EGRESSIP 맵을 캐시하도록 SSSD 구성

SSSD 서비스는 IdM 서버를 전혀 사용하도록 autofs 를 구성하지 않고도 IdM 서버에 저장된 autofs 맵을 캐시하는 데 사용할 수 있습니다.

사전 요구 사항

- sssd 패키지가 설치되어 있습니다.

절차

1. SSSD 구성 파일을 엽니다.

```
# vim /etc/sssd/sssd.conf
```

2.

SSSD 에서 처리하는 서비스 목록에 EgressIP 서비스를 추가합니다.

```
[sssd]
domains = ldap
services = nss,pam,autofs
```

3.

새 [autofs] 섹션을 만듭니다. EgressIP 서비스의 기본 설정이 대부분의 인프라에서 작동하므로 이 공백 을 비워 둘 수 있습니다.

```
[nss]

[pam]

[sudo]

[autofs]

[ssh]

[pac]
```

자세한 내용은 sssd.conf 매뉴얼 페이지를 참조하십시오.

4.

필요한 경우 EgressIP 항목에 대한 검색 기반을 설정합니다. 기본적으로 LDAP 검색 기반이지만 ldap_autofs_search_base 매개변수에 하위 트리를 지정할 수 있습니다.

```
[domain/EXAMPLE]

ldap_search_base = "dc=example,dc=com"
ldap_autofs_search_base = "ou=automount,dc=example,dc=com"
```

5.

SSSD 서비스를 다시 시작하십시오.

```
# systemctl restart sssd.service
```

6.

SSSD가 자동 마운트 설정을 위한 소스로 나열되도록 /etc/nsswitch.conf 파일을 확인합니다.

```
automount: sss files
```

7. *EgressIP 서비스를 다시 시작하십시오.*

```
# systemctl restart autofs.service
```

8. */home에 대한 마스터 맵 항목이 있다고 가정하여 사용자의 /home 디렉토리를 나열하여 구성을 테스트합니다.*

```
# ls /home/userName
```

원격 파일 시스템을 마운트하지 않으면 /var/log/message 파일에 오류가 있는지 확인하십시오. 필요한 경우 logging 매개 변수를 debug 로 설정하여 /etc/sysconfig/autofs 파일의 디버그 수준을 높입니다.

34장. 루트 파일 시스템에 대한 읽기 전용 권한 설정

읽기 전용 권한으로 루트 파일 시스템(/)을 마운트해야 하는 경우가 있습니다. 예제 사용 사례는 예기치 않은 시스템 전원 끄기 후 보안을 강화하거나 데이터 무결성을 보장하는 것입니다.

34.1. 쓰기 권한을 항상 유지하는 파일 및 디렉터리

시스템이 제대로 작동하려면 일부 파일과 디렉터리가 쓰기 권한을 유지해야 합니다. 루트 파일 시스템이 읽기 전용 모드로 마운트되면 이러한 파일은 **tmpfs** 임시 파일 시스템을 사용하여 **RAM**에 마운트됩니다.

이러한 파일 및 디렉터리의 기본 세트는 **/etc/rwtab** 파일에서 읽습니다. 시스템에 이 파일이 있어야 하는 경우 **readonly-root** 패키지가 필요합니다.

```
dirs /var/cache/man
dirs /var/gdm
<content truncated>

empty /tmp
empty /var/cache/foomatic
<content truncated>

files /etc/adjtime
files /etc/ntp.conf
<content truncated>
```

/etc/rwtab 파일의 항목은 다음 형식을 따릅니다.

```
copy-method    path
```

이 구문에서 다음을 수행합니다.

- 파일 또는 디렉터리가 **tmpfs**에 복사되는 방법을 지정하는 키워드 중 하나로 **copy-method** 를 교체합니다.
- **path** 를 파일 또는 디렉터리의 경로로 바꿉니다.

/etc/rwtab 파일은 파일 또는 디렉토리를 **tmpfs** 에 복사할 수 있는 다음과 같은 방법을 인식합니다.

empty

빈 경로가 **tmpfs** 에 복사됩니다. 예를 들면 다음과 같습니다.

```
empty /tmp
```

dirs

디렉터리 트리가 **tmpfs,** 비어 있습니다. 예를 들면 다음과 같습니다.

```
dirs /var/run
```

파일

파일 또는 디렉터리 트리는 그대로 **tmpfs** 에 복사됩니다. 예를 들면 다음과 같습니다.

```
files /etc/resolv.conf
```

/etc/rwtab.d/ 에 사용자 지정 경로를 추가할 때 동일한 형식이 적용됩니다.

34.2. 부팅 시 읽기 전용 권한으로 마운트하도록 루트 파일 시스템 구성

이 절차를 수행하면 다음 모든 부팅 시 루트 파일 시스템이 읽기 전용으로 마운트됩니다.

절차

1. **/etc/sysconfig/readonly-root** 파일에서 **READONLY** 옵션을 **yes** 로 설정합니다.

```
# Set to 'yes' to mount the file systems as read-only.
READONLY=yes
```

2. **/etc/fstab** 파일의 루트 항목(/)에 **ro** 옵션을 추가합니다.

```
/dev/mapper/luks-c376919e... / xfs x-systemd.device-timeout=0,ro 1 1
```

3. **ro** 커널 옵션을 활성화합니다.

```
# grubby --update-kernel=ALL --args="ro"
```

4. ***rw** 커널 옵션이 비활성화되어 있는지 확인합니다.*

```
# grubby --update-kernel=ALL --remove-args="rw"
```

5. ***tmpfs** 파일 시스템에서 쓰기 권한을 사용하여 마운트할 파일과 디렉토리를 추가해야 하는 경우 **/etc/rwtab.d/** 디렉터리에 텍스트 파일을 생성하고 해당 디렉토리에 구성을 배치합니다.*

*예를 들어 쓰기 권한으로 **/etc/example/file** 파일을 마운트하려면 **/etc/rwtab.d/example** 파일에 다음 행을 추가합니다.*

```
files /etc/example/file
```

중요

***tmpfs** 의 파일과 디렉터리에 대한 변경 사항은 부팅 시 유지되지 않습니다.*

6. *시스템을 재부팅하여 변경 사항을 적용합니다.*

문제 해결

- *읽기 전용 권한으로 루트 파일 시스템을 잘못 마운트하면 다음 명령을 사용하여 읽기 및 쓰기 권한으로 다시 마운트할 수 있습니다.*

```
# mount -o remount,rw /
```

35장. 할당량으로 XFS의 스토리지 공간 사용 제한

디스크 할당량을 구현하여 사용자 또는 그룹에서 사용할 수 있는 디스크 공간을 제한할 수 있습니다. 사용자가 너무 많은 디스크 공간을 사용하거나 파티션이 가득 차기 전에 시스템 관리자에게 알리는 경고 수준을 정의할 수도 있습니다.

XFS 할당량 하위 시스템은 디스크 공간(블록) 및 파일(**inode**) 사용량에 대한 제한을 관리합니다. **XFS** 할당량은 사용자, 그룹 또는 디렉터리 또는 프로젝트 수준에서 이러한 항목을 사용하거나 사용합니다. 그룹 및 프로젝트 할당량은 이전 기본이 아닌 **XFS** 디스크 형식에서만 상호 배타적입니다.

디렉토리별 또는 프로젝트별로 관리할 때 **XFS**는 특정 프로젝트와 관련된 디렉터리 계층의 디스크 사용을 관리합니다.

35.1. 디스크 할당량

대부분의 컴퓨팅 환경에서는 디스크 공간이 무한하지 않습니다. 할당량 하위 시스템은 디스크 공간의 사용을 제어하는 메커니즘을 제공합니다.

로컬 파일 시스템에서 개별 사용자 및 사용자 그룹에 대한 디스크 할당량을 구성할 수 있습니다. 이를 통해 사용자별 파일(예: 이메일)에 할당된 공간을 사용자가 작업하는 프로젝트에 할당된 공간과 별도로 관리할 수 있습니다. 할당량 하위 시스템은 할당된 제한을 초과할 때 사용자에게 경고하지만 현재 작업(하드 제한/소프트 제한)에 대해 일부 추가 공간을 허용합니다.

할당량을 구현하는 경우 할당량이 초과되었는지 확인하고 할당량이 올바른지 확인해야 합니다. 사용자가 할당량을 반복적으로 초과하거나 소프트 제한에 지속적으로 도달하는 경우 시스템 관리자는 사용자가 디스크 공간을 적게 사용하는 방법을 결정하거나 사용자의 디스크 할당량을 늘리는 데 도움이 될 수 있습니다.

제어할 할당량을 설정할 수 있습니다.

- 사용된 디스크 블록 수입니다.
- **UNIX** 파일 시스템의 파일에 대한 정보가 포함된 데이터 구조인 **inode** 수입니다. **inode**는 파일 관련 정보를 저장하므로 생성할 수 있는 파일 수를 제어할 수 있습니다.

35.2. XFS_QUOTA 툴

xfs_quota 툴을 사용하여 **XFS** 파일 시스템의 할당량을 관리할 수 있습니다. 또한 제한 적용이 설정된 **XFS** 파일 시스템을 유효 디스크 사용 회계 시스템으로 사용할 수 있습니다.

XFS 할당량 시스템은 여러 가지 면에서 다른 파일 시스템과 다릅니다. 가장 중요한 점은 **XFS**는 할당량 정보를 파일 시스템 메타데이터로 간주하고 저널링을 사용하여 일관성을 높은 수준으로 보장합니다.

추가 리소스

- **xfs_quota(8)** 도움말 페이지.

35.3. XFS의 파일 시스템 할당량 관리

XFS 할당량 하위 시스템은 디스크 공간(블록) 및 파일(**inode**) 사용량에 대한 제한을 관리합니다. **XFS** 할당량은 사용자, 그룹 또는 디렉터리 또는 프로젝트 수준에서 이러한 항목을 사용하거나 사용합니다. 그룹 및 프로젝트 할당량은 이전 기본이 아닌 **XFS** 디스크 형식에서만 상호 배타적입니다.

디렉토리별 또는 프로젝트별로 관리할 때 **XFS**는 특정 프로젝트와 관련된 디렉터리 계층의 디스크 사용을 관리합니다.

35.4. XFS의 디스크 할당량 활성화

이 절차에서는 **XFS** 파일 시스템에서 사용자, 그룹 및 프로젝트에 대한 디스크 할당량을 활성화합니다. 할당량이 활성화되면 **xfs_quota** 툴을 사용하여 제한을 설정하고 디스크 사용량을 보고할 수 있습니다.

절차

1. 사용자에 대한 할당량을 활성화합니다.

```
# mount -o uquota /dev/xvdb1 /xfs
```

uquota 를 **uqnoenforce** 로 교체하여 제한 없이 사용 보고를 허용합니다.

2. 그룹에 대한 할당량을 활성화합니다.

```
# mount -o gquota /dev/xvdb1 /xfs
```

제한 없이 사용 보고를 허용하도록 **gquota** 를 **gqnoenforce** 로 바꿉니다.

3. 프로젝트의 할당량을 활성화합니다.

```
# mount -o pquota /dev/xvdb1 /xfs
```

제한 없이 사용 보고를 허용하도록 **pquota** 를 **pqnoenforce** 로 바꿉니다.

4. 또는 **/etc/fstab** 파일에 할당량 마운트 옵션을 포함합니다. 다음 예제에서는 **XFS** 파일 시스템에서 각각 사용자, 그룹 및 프로젝트에 대한 할당량을 활성화하는 **/etc/fstab** 파일의 항목을 보여줍니다. 다음 예제에서는 읽기/쓰기 권한을 사용하여 파일 시스템을 마운트합니다.

```
# vim /etc/fstab
/dev/xvdb1   /xfs   xfs   rw,quota      0  0
/dev/xvdb1   /xfs   xfs   rw,gquota     0  0
/dev/xvdb1   /xfs   xfs   rw,prjquota   0  0
```

추가 리소스

- **mount(8)** 도움말 페이지.
- **xfs_quota(8)** 도움말 페이지.

35.5. XFS 사용량 보고

xfs_quota 툴을 사용하여 디스크 사용량에 대한 제한 및 보고서를 설정할 수 있습니다. 기본적으로 **xfs_quota** 는 대화식으로 실행되며 기본 모드에서 실행됩니다. 기본 모드 하위 명령은 단순히 사용법을 보고하며 모든 사용자가 사용할 수 있습니다.

사전 요구 사항

- **XFS** 파일 시스템에 대한 할당량이 활성화되었습니다. **XFS용 디스크 할당량 활성화**를 참조하십시오.

절차

1. **xfs_quota** 쉘을 시작합니다.

```
# xfs_quota
```

2. 지정된 사용자에 대한 사용량 및 제한을 표시합니다.

```
# xfs_quota> quota username
```

3. 블록 및 **inode**에 대해 자유 및 사용된 개수를 표시합니다.

```
# xfs_quota> df
```

4. **help** 명령을 실행하여 **xfs_quota** 에서 사용할 수 있는 기본 명령을 표시합니다.

```
# xfs_quota> help
```

5. **xfs_quota** 를 종료하려면 **q** 를 지정합니다.

```
# xfs_quota> q
```

추가 리소스

- **xfs_quota(8)** 도움말 페이지.

35.6. XFS 할당량 제한 수정

-x 옵션으로 **xfs_quota** 도구를 시작하여 전문가 모드를 활성화하고 관리자 명령을 실행하여 할당량 시스템을 수정할 수 있습니다. 이 모드의 하위 명령은 실제 제한을 구성할 수 있으며 상승된 권한이 있는 사용자만 사용할 수 있습니다.

사전 요구 사항

- **XFS** 파일 시스템에 대한 할당량이 활성화되었습니다. *XFS용 디스크 할당량 활성화*를 참조하십시오.

절차

1. ***-x** 옵션으로 **xfs_quota** 쉘을 시작하여 전문가 모드를 활성화합니다.*

```
# xfs_quota -x
```

2. *특정 파일 시스템의 할당량 정보를 보고합니다.*

```
# xfs_quota> report /path
```

*예를 들어 **/home (/dev/blockdevice)**에 대한 샘플 할당량 보고서를 표시하려면 명령 **report -h /home** 을 사용합니다. 다음과 유사한 출력이 표시됩니다.*

```
User quota on /home (/dev/blockdevice)
Blocks
User ID      Used   Soft   Hard Warn/Grace
---------- --------------------------------
root            0      0      0  00 [------]
testuser   103.4G      0      0  00 [------]
```

3. *할당량 제한을 수정합니다.*

```
# xfs_quota> limit isoft=500m ihard=700m user /path
```

*예를 들어 홈 디렉터리가 **/home/john** 인 사용자 **john** 에 대해 각각 소프트 및 하드 **inode** 수 제한을 설정하려면 다음 명령을 사용하십시오.*

```
# xfs_quota -x -c 'limit isoft=500 ihard=700 john' /home/
```

*이 경우 마운트된 **xfs** 파일 시스템인 **mount_point** 를 전달합니다.*

4. ***help** 명령을 실행하여 **xfs_quota -x** 로 사용 가능한 전문 명령을 표시합니다.*

```
# xfs_quota> help
```

추가 리소스

- ***xfs_quota(8)** 도움말 페이지.*

35.7. XFS의 프로젝트 제한 설정

이 절차에서는 프로젝트 제어 디렉터리에 대한 제한을 구성합니다.

절차

1. 프로젝트 제어 디렉터리를 **/etc/projects** 에 추가합니다. 예를 들어 다음은 고유 **ID**가 **11**인 **/var/log** 경로를 **/etc/projects** 에 추가합니다. 프로젝트 **ID**는 프로젝트에 매핑된 모든 숫자 값일 수 있습니다.

```
# echo 11:/var/log >> /etc/projects
```

2. 프로젝트 이름을 **/etc/projid** 에 추가하여 프로젝트 **ID**를 프로젝트 이름에 매핑합니다. 예를 들어 다음에서는 이전 단계에서 정의한 대로 **logfiles** 라는 프로젝트를 프로젝트 **ID 11**과 연결합니다.

```
# echo logfiles:11 >> /etc/projid
```

3. 프로젝트 디렉터리를 초기화합니다. 예를 들어 다음은 프로젝트 디렉터리 **/var** 을 초기화합니다.

```
# xfs_quota -x -c 'project -s logfiles' /var
```

4. 초기화된 디렉터리를 사용하여 프로젝트의 할당량을 구성합니다.

```
# xfs_quota -x -c 'limit -p bhard=1g logfiles' /var
```

추가 리소스

- **xfs_quota(8)** 도움말 페이지.
- **projid(5)** 도움말 페이지.
- **projects(5)** 도움말 페이지.

36장. 할당량으로 EXT4의 스토리지 공간 사용 제한

시스템에서 디스크 할당량을 활성화해야 할당할 수 있습니다. 사용자당, 그룹당 또는 프로젝트당 디스크 할당량을 할당할 수 있습니다. 그러나 소프트 제한 세트가 있는 경우 유예 기간이라는 구성 가능한 기간 동안 이러한 할당량을 초과할 수 있습니다.

36.1. 할당량 도구 설치

디스크 할당량을 구현하려면 할당량 RPM 패키지를 설치해야 합니다.

절차

- quota 패키지를 설치합니다.

```
# dnf install quota
```

36.2. 파일 시스템 생성 시 할당량 기능 활성화

다음 절차에서는 파일 시스템 생성 시 할당량을 활성화하는 방법을 설명합니다.

절차

1. 파일 시스템 생성 시 할당량을 활성화합니다.

```
# mkfs.ext4 -O quota /dev/sda
```

참고

기본적으로 사용자 및 그룹 할당량만 활성화 및 초기화됩니다.

2. 파일 시스템 생성 시 기본값을 변경합니다.

```
# mkfs.ext4 -O quota -E quotatype=usrquota:grpquota:prjquota /dev/sda
```

3. 파일 시스템을 마운트합니다.

-

```
# mount /dev/sda
```

추가 리소스

- **ext4(5)** 도움말 페이지.

36.3. 기존 파일 시스템에서 할당량 기능 활성화

이 절차에서는 **tune2fs** 명령을 사용하여 기존 파일 시스템에서 할당량 기능을 활성화하는 방법을 설명합니다.

절차

1. 파일 시스템을 마운트 해제합니다.

```
# umount /dev/sda
```

2. 기존 파일 시스템에서 할당량을 활성화합니다.

```
# tune2fs -O quota /dev/sda
```

참고

기본적으로 사용자 및 그룹 할당량만 초기화됩니다.

3. 기본값을 변경합니다.

```
# tune2fs -Q usrquota,grpquota,prjquota /dev/sda
```

4. 파일 시스템을 마운트합니다.

```
# mount /dev/sda
```

추가 리소스

- **ext4(5)** 도움말 페이지.

36.4. 할당량 적용 활성화

추가 옵션 없이 파일 시스템을 마운트한 후에는 기본적으로 할당량 계정이 활성화되어 있지만 할당량 적용은 적용되지 않습니다.

사전 요구 사항

- 할당량 기능이 활성화되고 기본 할당량이 초기화됩니다.

절차

- 사용자 할당량에 대한 **quotaon** 을 통해 할당량을 활성화합니다.

```
# mount /dev/sda /mnt
```

```
# quotaon /mnt
```

참고

할당량 적용은 **${quota ,grp quota** 또는 **prjquota** 마운트 옵션을 사용하여 마운트 시 활성화할 수 있습니다.

```
# mount -o usrquota,grpquota,prjquota /dev/sda /mnt
```

- 모든 파일 시스템에 대해 사용자, 그룹 및 프로젝트 할당량을 활성화합니다.

```
# quotaon -vaugP
```

  - **-u,-g** 또는 **-P** 옵션이 모두 지정되지 않은 경우 사용자 할당량만 활성화됩니다.
  - **-g** 옵션만 지정하면 그룹 할당량만 활성화됩니다.
  - **-P** 옵션만 지정하면 프로젝트 할당량만 활성화됩니다.

- **/home** 과 같은 특정 파일 시스템에 대한 할당량을 활성화합니다.

```
# quotaon -vugP /home
```

추가 리소스

- **quotaon(8)** 도움말 페이지.

36.5. 사용자당 할당량 할당

디스크 할당량은 **edquota** 명령을 사용하여 사용자에게 할당됩니다.

참고

EDITOR 환경 변수에서 정의한 텍스트 편집기는 **edquota** 에서 사용됩니다. 편집기를 변경하려면 **~/.bash_profile** 파일의 **EDITOR** 환경 변수를 선택한 편집기의 전체 경로로 설정합니다.

사전 요구 사항

- 사용자 할당량을 설정하기 전에 사용자가 존재해야 합니다.

절차

1. 사용자 할당량을 할당합니다.

```
# edquota username
```

할당량을 할당할 사용자로 사용자 이름을 교체합니다.

예를 들어 **/dev/sda** 파티션에 대한 할당량을 활성화하고 **edquota testuser** 명령을 실행하면 시스템에 구성된 기본 편집기에 다음이 표시됩니다.

```
Disk quotas for user testuser (uid 501):
Filesystem   blocks   soft   hard   inodes   soft   hard
/dev/sda      44043      0      0   37418      0      0
```

2. 원하는 제한 사항을 변경합니다.

*값 중 하나를 **0**으로 설정하면 **limit**이 설정되지 않습니다. 텍스트 편집기에서 변경합니다.*

*예를 들어, 다음은 **testuser**의 소프트 및 하드 블록 제한이 각각 **50000** 및 **55000**으로 설정되어 있음을 보여줍니다.*

```
Disk quotas for user testuser (uid 501):
Filesystem   blocks   soft   hard   inodes   soft   hard
/dev/sda      44043  50000  55000    37418      0      0
```

- *첫 번째 열은 할당량이 활성화된 파일 시스템의 이름입니다.*
- *두 번째 열에는 현재 사용자가 사용 중인 블록 수가 표시됩니다.*
- *다음 두 열은 파일 시스템에서 사용자에 대한 소프트 및 하드 블록 제한을 설정하는 데 사용됩니다.*
- ***inodes** 열에는 현재 사용자가 사용 중인 **inode** 수가 표시됩니다.*
- *마지막 두 열은 파일 시스템에서 사용자의 소프트 및 하드 **inode** 제한을 설정하는 데 사용됩니다.*
  - *하드 블록 제한은 사용자 또는 그룹이 사용할 수 있는 디스크 공간의 절대 최대 크기입니다. 이 제한에 도달하면 추가 디스크 공간을 사용할 수 없습니다.*
  - *소프트 블록 제한은 사용할 수 있는 최대 디스크 공간을 정의합니다. 그러나 하드 제한과 달리 소프트 제한은 일정 시간 동안 초과될 수 있습니다. 이 시간을 **grace period**라고 합니다. 유예 기간은 초, 분, 시간, 일, 주 또는 개월로 나타낼 수 있습니다.*

검증 단계

- *사용자 할당량이 설정되었는지 확인합니다.*

```
# quota -v testuser
Disk quotas for user testuser:
Filesystem  blocks  quota  limit  grace  files  quota  limit  grace
/dev/sda     1000*  1000   1000           0      0      0
```

36.6. 그룹당 할당량 할당

그룹별로 할당량을 할당할 수 있습니다.

사전 요구 사항

- 그룹 할당량을 설정하기 전에 **group**이 있어야 합니다.

절차

1. 그룹 할당량을 설정합니다.

```
# edquota -g groupname
```

예를 들어 **devel** 그룹에 그룹 할당량을 설정하려면 다음을 수행합니다.

```
# edquota -g devel
```

이 명령은 텍스트 편집기에서 그룹의 기존 할당량을 표시합니다.

```
Disk quotas for group devel (gid 505):
Filesystem   blocks  soft  hard  inodes  soft  hard
/dev/sda     440400     0     0   37418     0     0
```

2. 제한을 수정하고 파일을 저장합니다.

검증 단계

- 그룹 할당량이 설정되어 있는지 확인합니다.

```
# quota -vg groupname
```

36.7. 프로젝트당 할당량 할당

이 절차에서는 프로젝트당 할당량을 할당합니다.

사전 요구 사항

- *파일 시스템에서 프로젝트 할당량이 활성화되어 있습니다.*

절차

1. *프로젝트 제어 디렉터리를 **/etc/projects** 에 추가합니다. 예를 들어 다음은 고유 **ID**가 **11**인 **/var/log** 경로를 **/etc/projects** 에 추가합니다. 프로젝트 **ID**는 프로젝트에 매핑된 모든 숫자 값일 수 있습니다.*

```
# echo 11:/var/log >> /etc/projects
```

2. *프로젝트 이름을 **/etc/projid** 에 추가하여 프로젝트 **ID**를 프로젝트 이름에 매핑합니다. 예를 들어, 다음에서는 이전 단계에서 정의한 대로 **Logs** 라는 프로젝트와 프로젝트 **ID 11**을 연결합니다.*

```
# echo Logs:11 >> /etc/projid
```

3. *원하는 제한을 설정합니다.*

```
# edquota -P 11
```

참고

*프로젝트 **ID**(이 경우**11**) 또는 해당 이름(이 경우로그)으로 프로젝트를 선택할 수 있습니다.*

4. ***quotaon** 을 사용하여 할당량 적용을 활성화합니다.*

할당량 적용 활성화를 참조하십시오.

검증 단계

- *프로젝트 할당량이 설정되어 있는지 확인합니다.*

```
# quota -vP 11
```

참고

프로젝트 ID 또는 프로젝트 이름으로 확인할 수 있습니다.

추가 리소스

- edquota(8) 도움말 페이지.
- projid(5) 도움말 페이지.
- projects(5) 도움말 페이지.

36.8. 소프트 제한의 유예 기간 설정

지정된 할당량에 소프트 제한이 있는 경우 소프트 제한을 초과할 수 있는 기간인 유예 기간을 편집할 수 있습니다. 사용자, 그룹 또는 프로젝트에 대한 유예 기간을 설정할 수 있습니다.

절차

- 유예 기간을 편집합니다.

```
# edquota -t
```

중요

다른 edquota 명령은 특정 사용자, 그룹 또는 프로젝트의 할당량에서 작동하지만 -t 옵션은 할당량이 활성화된 모든 파일 시스템에서 작동합니다.

추가 리소스

- edquota(8) 도움말 페이지.

36.9. 파일 시스템 할당량 비활성화

할당량off 를 사용하여 지정된 파일 시스템에서 디스크 할당량 적용을 끕니다. 이 명령을 실행한 후 할당량 계정이 활성화된 상태로 유지됩니다.

절차

- *모든 사용자 및 그룹 할당량을 활성화하려면 다음을 수행합니다.*

```
# quotaoff -vaugP
```

  - ***-u,-g** 또는 **-P** 옵션이 모두 지정되지 않은 경우 사용자 할당량만 비활성화됩니다.*
  - ***-g** 옵션만 지정하면 그룹 할당량만 비활성화됩니다.*
  - ***-P** 옵션만 지정하면 프로젝트 할당량만 비활성화됩니다.*
  - ***-v** 스위치를 사용하면 명령이 실행될 때 자세한 상태 정보가 표시됩니다.*

추가 리소스

- ***quotaoff(8)** 도움말 페이지.*

36.10. 디스크 할당량 보고

***repquota** 유틸리티를 사용하여 디스크 할당량 보고서를 생성할 수 있습니다.*

절차

1. ***repquota** 명령을 실행합니다.*

```
# repquota
```

   *예를 들어 **repquota /dev/sda** 명령은 다음 출력을 생성합니다.*

```
*** Report for user quotas on device /dev/sda
Block grace time: 7days; Inode grace time: 7days
  Block limits   File limits
User  used soft hard grace used soft hard grace
```

```
----------------------------------------------------------------------
root      --      36       0       0           4     0     0
kristin   --     540       0       0         125     0     0
testuser  --  440400  500000  550000       37418     0     0
```

2. 모든 할당량 활성화 파일 시스템에 대한 디스크 사용량 보고서를 확인합니다.

```
# repquota -augP
```

각 사용자가 블록 또는 **inode** 제한을 초과했는지 여부를 확인한 후 **--** 기호가 표시됩니다. 소프트 제한이 두 개 이상되면 해당 **-** 문자 대신 **+** 문자가 표시됩니다. 첫 번째 **-** 문자는 블록 제한을 나타내며, 두 번째 문자는 **inode** 제한을 나타냅니다.

일반적으로 유예 열은 비어 있습니다. 소프트 제한이 초과된 경우 열에 유예 기간에 남은 시간 크기와 같은 시간 사양이 포함됩니다. 유예 기간이 만료되면 그 자리에 아무것도 나타나지 않습니다.

추가 리소스

자세한 내용은 **repquota(8)** 도움말 페이지.

37장. 사용되지 않는 블록 삭제

이를 지원하는 블록 장치에서 삭제 작업을 수행하거나 예약할 수 있습니다.

37.1. 블록 삭제 작업

블록 삭제 작업은 마운트된 파일 시스템에서 더 이상 사용되지 않는 블록을 삭제합니다. 다음 작업에 유용합니다.

- SSD(솔리드 스테이트 드라이브)
- 씬 프로비저닝된 스토리지

요구 사항

파일 시스템의 기본 블록 장치는 물리적 삭제 작업을 지원해야 합니다.

/sys/block/장치/queue/discard_max_bytes 파일의 값이 0이 아닌 경우 물리적 삭제 작업이 지원됩니다.

37.2. 블록 삭제 작업 유형

다른 방법을 사용하여 삭제 작업을 실행할 수 있습니다.

배치 삭제

사용자가 명시적으로 실행합니다. 선택한 파일 시스템에서 사용되지 않는 모든 블록을 삭제합니다.

온라인 삭제

마운트 시 지정합니다. 사용자 개입 없이 실시간으로 실행됩니다. 온라인 삭제 작업은 사용되는 블록에서 자유로 전환되는 블록만 삭제합니다.

주기적 삭제

systemd 서비스에서 정기적으로 실행되는 배치 작업입니다.

모든 유형은 **XFS** 및 **ext4** 파일 시스템과 **VDO**에서 지원됩니다.

권장 사항

배치 또는 정기적인 삭제를 사용하는 것이 좋습니다.

온라인 삭제는 다음과 같은 경우에만 사용하십시오.

- 일괄 삭제가 불가능하거나 시스템의 워크로드입니다.
- 온라인 삭제 작업은 성능을 유지하기 위해 필요합니다.

37.3. 배치 블록 삭제 수행

다음 절차에서는 배치 블록 삭제 작업을 수행하여 마운트된 파일 시스템에서 사용되지 않은 블록을 삭제합니다.

사전 요구 사항

- 파일 시스템이 마운트됩니다.
- 파일 시스템 기반이 되는 블록 장치는 물리적 삭제 작업을 지원합니다.

절차

- **fstrim** 유틸리티를 사용합니다.
  - 선택한 파일 시스템에서만 삭제 작업을 수행하려면 다음을 사용하십시오.

```
# fstrim mount-point
```

  - 마운트된 모든 파일 시스템에서 삭제 작업을 수행하려면 다음을 사용하십시오.

```
# fstrim --all
```

다음에서 **fstrim** 명령을 실행하는 경우:

- 삭제 작업을 지원하지 않는 장치 또는
- 장치 중 하나가 삭제 작업을 지원하지 않는 여러 장치로 구성된 논리 장치(**LVM** 또는 **MD**)

다음 메시지가 표시됩니다.

```
# fstrim /mnt/non_discard

fstrim: /mnt/non_discard: the discard operation is not supported
```

추가 리소스

- **fstrim(8)** 도움말 페이지.

37.4. 온라인 블록 삭제 활성화

이 절차에서는 지원되는 모든 파일 시스템에서 사용되지 않은 블록을 자동으로 삭제하는 온라인 블록 삭제 작업을 활성화합니다.

절차

- 마운트 시 온라인 삭제 활성화:
  - 파일 시스템을 수동으로 마운트하는 경우 **-o** 삭제 마운트 옵션을 추가합니다.

    ```
    # mount -o discard device mount-point
    ```

  - 파일 시스템을 영구적으로 마운트할 때 **/etc/fstab** 파일의 마운트 항목에 삭제 옵션을 추가합니다.

추가 리소스

- ***mount(8)** 도움말 페이지.*

- ***fstab(5)** 도움말 페이지.*

37.5. 주기적 블록 삭제 활성화

*이 절차에서는 지원되는 모든 파일 시스템에서 사용되지 않는 블록을 정기적으로 삭제하는 **systemd** 타이머를 활성화합니다.*

절차

- ***systemd** 타이머를 활성화하고 시작합니다.*

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

38장. RHEL 시스템 역할을 사용하여 온라인 블록 삭제 활성화

이 섹션에서는 스토리지 역할을 사용하여 온라인 블록 삭제 기능을 활성화하는 방법을 설명합니다.

사전 요구 사항

- 스토리지 역할을 포함한 Ansible 플레이북이 있습니다.

38.1. 온라인 블록 삭제 활성화를 위한 ANSIBLE PLAYBOOK 예

이 섹션에서는 예제 Ansible 플레이북을 제공합니다. 이 Playbook은 온라인 블록 삭제가 활성화된 XFS 파일 시스템을 마운트하는 스토리지 역할을 적용합니다.

예 38.1. /mnt/data/에서 온라인 블록 삭제를 활성화하는 플레이북

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
        mount_options: discard
  roles:
    - rhel-system-roles.storage
```

추가 리소스

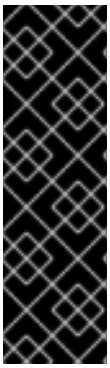
- Ansible 플레이북의 예제로 파일 시스템을 지속적으로 마운트합니다.
- /usr/share/ansible/roles/rhel-system-roles.storage/README.md 파일.

38.2. 추가 리소스

- 파일 시스템 관리.

39장. STRATIS 파일 시스템 설정

Stratis는 물리적 스토리지 장치 풀을 관리하기 위해 서비스로 실행되며, 복잡한 스토리지 구성을 설정하고 관리할 수 있도록 로컬 스토리지 관리를 단순화합니다.



중요

Stratis는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 **Red Hat** 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. **Red Hat**은 프로덕션 환경에서 사용하지 않는 것이 좋습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다. **Red Hat** 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 https://access.redhat.com/support/offerings/techpreview 에서 참조하십시오.

39.1. STRATIS란 무엇입니까?

Stratis는 **Linux**용 로컬 스토리지 관리 솔루션입니다. 단순성과 사용 편의성에 중점을 두고 고급 스토리지 기능에 액세스할 수 있습니다.

Stratis는 다음 작업을 더 쉽게 수행할 수 있습니다.

- 스토리지의 초기 구성
- 나중에 변경 내용
- 고급 스토리지 기능 사용

Stratis는 고급 스토리지 기능을 지원하는 하이브리드 사용자 및 커널 로컬 스토리지 관리 시스템입니다. **Stratis**의 주요 개념은 스토리지 풀 입니다. 이 풀은 하나 이상의 로컬 디스크 또는 파티션에서 생성되며 풀에서 볼륨이 생성됩니다.

이 풀을 사용하면 다음과 같은 다양한 유용한 기능을 사용할 수 있습니다.

- 파일 시스템 스냅샷

- 씬 프로비저닝
- 계층화

추가 리소스

- Stratis 웹 사이트

39.2. STRATIS 볼륨의 구성 요소

Stratis 볼륨을 구성하는 구성 요소에 대해 알아봅니다.

외부에서 Stratis는 명령줄 인터페이스 및 API에 다음 볼륨 구성 요소를 제공합니다.

blockdev

디스크 또는 디스크 파티션과 같은 블록 장치.

pool

하나 이상의 블록 장치로 구성됩니다.

풀은 블록 장치의 크기와 같은 고정된 전체 크기를 갖습니다.

이 풀에는 dm-cache 대상을 사용하는 비휘발성 데이터 캐시와 같은 대부분의 Stratis 계층이 포함되어 있습니다.

Stratis는 각 풀에 대해 /dev/stratis/my-pool/ 디렉터리를 생성합니다. 이 디렉터리에는 풀의 Stratis 파일 시스템을 나타내는 장치에 대한 링크가 포함되어 있습니다.

filesystem

각 풀에는 파일을 저장하는 하나 이상의 파일 시스템이 포함될 수 있습니다.

파일 시스템은 씬 프로비저닝되며 전체 크기가 고정되어 있지 않습니다. 파일 시스템의 실제 크기는 저장된 데이터와 함께 증가합니다. 데이터 크기가 파일 시스템의 가상 크기에 도달하면 Stratis에서

씬 볼륨과 파일 시스템을 자동으로 확장합니다.

파일 시스템은 XFS로 포맷됩니다.

중요

Stratis는 Stratis를 사용하여 생성된 파일 시스템에 대한 정보를 추적하며, XFS를 사용하여 변경한 내용은 Stratis에서 자동으로 업데이트되지 않습니다. Stratis에서 관리하는 XFS 파일 시스템을 다시 포맷하거나 재구성해서는 안 됩니다.

Stratis는 /dev/stratis/my-pool/my-fs 경로에 파일 시스템에 대한 링크를 생성합니다.

참고

Stratis는 dmsetup 목록 및 /proc/partitions 파일에 표시되는 많은 장치 매퍼 장치를 사용합니다. 마찬가지로 lsblk 명령 출력은 Stratis의 내부 작업 및 계층을 반영합니다.

39.3. STRATIS에서 사용할 수 있는 블록 장치

Stratis와 함께 사용할 수 있는 스토리지 장치.

지원되는 장치

Stratis 풀은 이러한 유형의 블록 장치에서 작동하도록 테스트되었습니다.

- LUKS
- LVM 논리 볼륨
- MD RAID
- DM Multipath

- **iSCSI**
- **HDD 및 SSD**
- **NVMe 장치**

지원되지 않는 장치

Stratis에는 씬 프로비저닝 계층이 포함되어 있으므로 이미 씬 프로비저닝된 블록 장치에 **Stratis** 풀을 배치하지 않는 것이 좋습니다.

39.4. STRATIS 설치

Stratis에 필요한 패키지를 설치합니다.

절차

1. **Stratis** 서비스 및 명령줄 유틸리티를 제공하는 패키지를 설치합니다.

```
# dnf install stratisd stratis-cli
```

2. **stratisd** 서비스가 활성화되어 있는지 확인합니다.

```
# systemctl enable --now stratisd
```

39.5. 암호화되지 않은 STRATIS 풀 생성

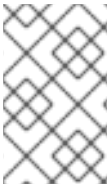
하나 이상의 블록 장치에서 암호화되지 않은 **Stratis** 풀을 생성할 수 있습니다.

사전 요구 사항

- **Stratis**가 설치되어 있습니다. 자세한 내용은 **Stratis** 설치를 참조하십시오.
- **stratisd** 서비스가 실행 중입니다.

- *Stratis 풀을 생성하는 블록 장치는 사용되지 않으며 마운트되지 않습니다.*
- *Stratis 풀을 생성하는 각 블록 장치는 최소 1GB입니다.*
- *IBM Z 아키텍처에서 /dev/dasd* 블록 장치를 파티셔닝해야 합니다. Stratis 풀에서 파티션을 사용합니다.*

DASD 장치를 파티셔닝 하는 방법에 대한 자세한 내용은 IBM Z에서 Linux 인스턴스 구성을 참조하십시오.



참고

암호화되지 않은 Stratis 풀을 암호화할 수 없습니다.

절차

1. *Stratis 풀에서 사용하려는 각 블록 장치에 존재하는 파일 시스템, 파티션 테이블 또는 RAID 서명을 삭제합니다.*

```
# wipefs --all block-device
```

여기서 block-device 는 블록 장치의 경로입니다(예: /dev/sdb).

2. *선택한 블록 장치에서 암호화되지 않은 새로운 Stratis 풀을 생성합니다.*

```
# stratis pool create my-pool block-device
```

여기서 block-device 는 비어 있거나 초기화된 블록 장치의 경로입니다.

참고

한 줄에 여러 블록 장치를 지정합니다.

```
# stratis pool create my-pool block-device-1 block-device-2
```

3. 새 Stratis 풀이 생성되었는지 확인합니다.

```
# stratis pool list
```

39.6. 암호화된 STRATIS 풀 생성

데이터를 보호하려면 하나 이상의 블록 장치에서 암호화된 Stratis 풀을 생성할 수 있습니다.

암호화된 Stratis 풀을 생성하면 커널 인증 키가 기본 암호화 메커니즘으로 사용됩니다. 후속 시스템을 재부팅한 후 이 커널 인증 키는 암호화된 Stratis 풀을 잠금 해제하는 데 사용됩니다.

하나 이상의 블록 장치에서 암호화된 Stratis 풀을 생성할 때 다음 사항에 유의하십시오.

- 각 블록 장치는 cryptsetup 라이브러리를 사용하여 암호화되며 LUKS2 형식을 구현합니다.
- 각 Stratis 풀에는 고유한 키를 포함하거나 다른 풀과 동일한 키를 공유할 수 있습니다. 이러한 키는 커널 인증 키에 저장됩니다.
- Stratis 풀을 구성하는 블록 장치는 암호화되거나 모든 암호화되지 않은 상태여야 합니다. 동일한 Stratis 풀에 암호화 및 암호화되지 않은 블록 장치를 둘 다 사용할 수 없습니다.
- 암호화된 Stratis 풀의 데이터 계층에 추가된 블록 장치는 자동으로 암호화됩니다.

사전 요구 사항

- Stratis v2.1.0 이상이 설치되었습니다. 자세한 내용은 Stratis 설치를 참조하십시오.
- stratisd 서비스가 실행 중입니다.
- Stratis 풀을 생성하는 블록 장치는 사용되지 않으며 마운트되지 않습니다.

- *Stratis 풀을 생성하는 블록 장치는 크기가 각각 1GB 이상입니다.*

- *IBM Z 아키텍처에서 /dev/dasd* 블록 장치를 파티셔닝해야 합니다. Stratis 풀에서 파티션을 사용합니다.*

DASD 장치를 파티셔닝 하는 방법에 대한 자세한 내용은 IBM Z에서 Linux 인스턴스 구성을 참조하십시오.

절차

1. *Stratis 풀에서 사용하려는 각 블록 장치에 존재하는 파일 시스템, 파티션 테이블 또는 RAID 서명을 삭제합니다.*

```
# wipefs --all block-device
```

여기서 block-device 는 블록 장치의 경로입니다(예: /dev/sdb).

2. *키 세트를 아직 생성하지 않은 경우 다음 명령을 실행하고 프롬프트에 따라 암호화에 사용할 키를 생성합니다.*

```
# stratis key set --capture-key key-description
```

여기서 key-description 은 커널 키링에서 생성되는 키에 대한 참조입니다.

3. *암호화된 Stratis 풀을 생성하고 암호화에 사용할 키 설명을 지정합니다. key-description 옵션을 사용하는 대신 --keyfile-path 옵션을 사용하여 키 경로를 지정할 수도 있습니다.*

```
# stratis pool create --key-desc key-description my-pool block-device
```

다음과 같습니다.

key-description

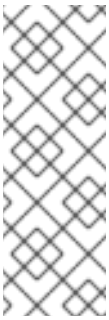
이전 단계에서 생성한 커널 인증 키에 존재하는 키를 참조합니다.

my-pool

*새 **Stratis** 풀의 이름을 지정합니다.*

block-device

비어 있거나 초기화된 블록 장치의 경로를 지정합니다.



참고

한 줄에 여러 블록 장치를 지정합니다.

```
# stratis pool create --key-desc key-description my-pool block-device-1
block-device-2
```

4. *새 **Stratis** 풀이 생성되었는지 확인합니다.*

```
# stratis pool list
```

39.7. STRATIS 파일 시스템에서 씬 프로비저닝 계층 설정

*스토리지 스택은 **overprovision** 상태에 도달할 수 있습니다. 파일 시스템 크기가 지원하는 풀보다 큰 경우 풀이 가득 차게 됩니다. 이를 방지하려면 과도한 프로비저닝을 비활성화하여 풀의 모든 파일 시스템의 크기가 풀에서 제공하는 사용 가능한 물리적 스토리지를 초과하지 않도록 합니다. 중요한 애플리케이션 또는 루트 파일 시스템에 **Stratis**를 사용하는 경우 이 모드에서는 특정 실패 사례를 방지합니다.*

*과도한 프로비저닝을 활성화하면 **API** 신호가 스토리지가 완전히 할당되었을 때 이를 알립니다. 알림은 모든 나머지 풀 공간이 가득 차면 **Stratis**에 확장할 공간이 없음을 알리는 경고 역할을 합니다.*

사전 요구 사항

- ***Stratis**가 설치되어 있습니다. 자세한 내용은 **Stratis** 설치를 참조하십시오.*

절차

풀을 올바르게 설정하려면 다음 두 가지 가능성이 있습니다.

1. *하나 이상의 블록 장치에서 풀을 생성합니다.*

```
# stratis pool create --no-overprovision pool-name /dev/sdb
```

- **no-overprovision** 옵션을 사용하면 풀에서 실제 사용 가능한 물리적 공간보다 더 많은 논리 공간을 할당할 수 없습니다.

2. 기존 풀에서 프로비저닝 모드로 설정합니다.

```
# stratis pool overprovision pool-name <yes|no>
```

- **"yes"**로 설정하면 풀에 과다 프로비저닝을 활성화합니다. 즉, **Stratis** 파일 시스템의 논리 크기 합계는 사용 가능한 데이터 공간 크기를 초과할 수 있습니다.

검증

1. 다음을 실행하여 **Stratis** 풀의 전체 목록을 확인합니다.

```
# stratis pool list

Name         Total Physical                Properties     UUID                                 Alerts
pool-name    1.42 TiB / 23.96 MiB / 1.42 TiB   ~Ca,~Cr,~Op    cb7cb4d8-9322-4ac4-a6fd-eb7ae9e1e540
```

2. **stratis pool list** 출력에 **overprovisioning mode** 플래그의 표시가 있는지 확인합니다. **"~"**은 **"NOT"**의 수학 기호이므로 **~Op** 는 **no-overprovisioning**을 의미합니다.

3. 선택 사항: 다음을 실행하여 특정 풀에서 과다 프로비저닝을 확인합니다.

```
# stratis pool overprovision pool-name yes

# stratis pool list

Name         Total Physical                Properties     UUID                                 Alerts
pool-name    1.42 TiB / 23.96 MiB / 1.42 TiB   ~Ca,~Cr,~Op    cb7cb4d8-9322-4ac4-a6fd-eb7ae9e1e540
```

추가 리소스

- **Stratis** 스토리지 웹 페이지.

39.8. STRATIS 풀 바인딩

암호화된 **Stratis** 풀을 **NBDE(Network Bound Disk Encryption)**에 바인딩하려면 **Tang** 서버가 필요합니다. **Stratis** 풀이 포함된 시스템이 재부팅되면 커널 키링 설명을 제공하지 않고도 암호화된 풀을 자동으로 잠금 해제하는 **Tang** 서버에 연결됩니다.

참고

Stratis 풀을 보조 **Clevis** 암호화 메커니즘에 바인딩해도 기본 커널 키링 암호화 메커니즘이 제거되지 않습니다.

사전 요구 사항

- **Stratis v2.3.0** 이상이 설치되어 있습니다. 자세한 내용은 **Stratis** 설치를 참조하십시오.
- **stratisd** 서비스가 실행 중입니다.
- 암호화된 **Stratis** 풀을 생성하고 암호화에 사용된 키에 대한 주요 설명이 있습니다. 자세한 내용은 암호화된 **Stratis** 풀 생성을 참조하십시오.
- **Tang** 서버에 연결할 수 있습니다. 자세한 내용은 강제 모드에서 **SELinux**를 사용하여 **Tang** 서버배포를 참조하십시오.

절차

- 암호화된 **Stratis** 풀을 **Clevis**에 바인딩합니다.

```
# stratis pool bind nbde --trust-url my-pool tang-server
```

다음과 같습니다.

my-pool

암호화된 **Stratis** 풀의 이름을 지정합니다.

tang-server

Tang 서버의 **IP** 주소 또는 **URL**을 지정합니다.

추가 리소스

- 정책 기반 암호 해독을 사용하여 암호화된 볼륨의 자동 잠금 해제 구성

39.9. STRATIS 풀을 TPM에 바인딩

암호화된 **Stratis** 풀을 신뢰할 수 있는 플랫폼 모듈(**TPM**) **2.0**에 바인딩하면 풀이 재부팅될 때 커널 인증 키링 설명을 제공하지 않아도 풀이 자동으로 잠금 해제됩니다.

사전 요구 사항

- **Stratis v2.3.0** 이상이 설치되어 있습니다. 자세한 내용은 **Stratis** 설치를 참조하십시오.
- **stratisd** 서비스가 실행 중입니다.
- 암호화된 **Stratis** 풀을 생성했습니다. 자세한 내용은 암호화된 **Stratis** 풀 생성을 참조하십시오.

절차

- 암호화된 **Stratis** 풀을 **TPM**에 바인딩합니다.

```
# stratis pool bind tpm my-pool key-description
```

다음과 같습니다.

my-pool

암호화된 **Stratis** 풀의 이름을 지정합니다.

key-description

암호화된 **Stratis** 풀을 생성할 때 생성된 커널 인증 키에 존재하는 키를 참조합니다.

39.10. 커널 인증 키를 사용하여 암호화된 STRATIS 풀 잠금 해제

시스템 재부팅 후 암호화된 **Stratis** 풀 또는 이를 구성하는 블록 장치가 표시되지 않을 수 있습니다. 풀을 암호화하는 데 사용된 커널 인증 키를 사용하여 풀의 잠금을 해제할 수 있습니다.

사전 요구 사항

- **Stratis v2.1.0**이 설치되어 있습니다. 자세한 내용은 **Stratis** 설치를 참조하십시오.
- **stratisd** 서비스가 실행 중입니다.
- 암호화된 **Stratis** 풀을 생성했습니다. 자세한 내용은 암호화된 **Stratis** 풀 생성을 참조하십시오.

절차

1. 이전에 사용된 것과 동일한 키 설명을 사용하여 키 세트를 다시 생성합니다.

```
# stratis key set --capture-key key-description
```

여기서 **key-description** 은 암호화된 **Stratis** 풀을 생성할 때 생성된 커널 인증 키에 있는 키를 참조합니다.

2. **Stratis** 풀 및 이를 구성하는 블록 장치의 잠금을 해제합니다.

```
# stratis pool unlock keyring
```

3. **Stratis** 풀이 표시되는지 확인합니다.

```
# stratis pool list
```

39.11. CLEVIS를 사용하여 암호화된 STRATIS 풀 잠금 해제

시스템 재부팅 후 암호화된 **Stratis** 풀 또는 이를 구성하는 블록 장치가 표시되지 않을 수 있습니다. 풀이 바인딩되는 보조 암호화 메커니즘을 사용하여 암호화된 **Stratis** 풀의 잠금을 해제할 수 있습니다.

사전 요구 사항

- ***Stratis v2.3.0** 이상이 설치되어 있습니다. 자세한 내용은 **Stratis** 설치를 참조하십시오.*
- ***stratisd** 서비스가 실행 중입니다.*
- *암호화된 **Stratis** 풀을 생성했습니다. 자세한 내용은 암호화된 **Stratis** 풀 생성을 참조하십시오.*
- *암호화된 **Stratis** 풀은 지원되는 보조 암호화 메커니즘에 바인딩됩니다. 자세한 내용은 암호화된 **Stratis** 풀 바인딩 또는 암호화된 **Stratis** 풀 바인딩을 **TPM** 에 참조하십시오.*

절차

1. ***Stratis** 풀 및 이를 구성하는 블록 장치의 잠금을 해제합니다.*

```
# stratis pool unlock clevis
```

2. ***Stratis** 풀이 표시되는지 확인합니다.*

```
# stratis pool list
```

39.12. 보조 암호화에서 STRATIS 풀 바인딩 해제

*지원되는 보조 암호화 메커니즘에서 암호화된 **Stratis** 풀을 바인딩 해제하면 기본 커널 키링 암호화 암호화가 그대로 유지됩니다.*

사전 요구 사항

- ***Stratis v2.3.0** 이상이 시스템에 설치되어 있습니다. 자세한 내용은 **Stratis** 설치를 참조하십시오.*
- *암호화된 **Stratis** 풀을 생성했습니다. 자세한 내용은 암호화된 **Stratis** 풀 생성을 참조하십시오.*

- 암호화된 **Stratis** 풀은 지원되는 보조 암호화 메커니즘에 바인딩됩니다.

절차

- 암호화된 **Stratis** 풀을 보조 암호화 메커니즘에서 바인딩 해제합니다.

```
# stratis pool unbind clevis my-pool
```

다음과 같습니다.

my-pool 은 바인딩 해제하려는 **Stratis** 풀의 이름을 지정합니다.

추가 리소스

- 암호화된 **Stratis** 풀을 **KnativeServing**에 바인딩
- 암호화된 **Stratis** 풀을 **TPM**에 바인딩

39.13. STRATIS 풀 시작 및 중지

Stratis 풀을 시작하고 중지할 수 있습니다. 이를 통해 파일 시스템, 캐시 장치, 씬 풀 및 암호화된 장치와 같은 풀을 구성하는 데 사용된 모든 개체를 분리하거나 가져올 수 있습니다. 풀이 적극적으로 모든 장치 또는 파일 시스템을 사용하는 경우 경고를 발행하고 중지할 수 없습니다.

중지된 풀은 해당 메타데이터에 중지된 상태를 기록합니다. 이러한 풀은 풀에서 시작 명령을 수신할 때까지 다음 부팅 시 시작되지 않습니다.

암호화되지 않은 경우 이전에 시작된 풀은 부팅 시 자동으로 시작됩니다. 풀 잠금 해제는 이 **Stratis** 버전에서 시작되는 풀 로 교체되므로 암호화된 풀 은 부팅 시 항상 **pool start** 명령이 필요합니다.

사전 요구 사항

- **Stratis**가 설치되어 있습니다. 자세한 내용은 **Stratis** 설치를 참조하십시오.

- ***stratisd** 서비스가 실행 중입니다.*

- *암호화되지 않은 또는 암호화된 **Stratis** 풀 중 하나를 생성했습니다. 암호화되지 않은 **Stratis** 풀 생성 또는 암호화된 **Stratis** 풀 생성을 참조하십시오.*

절차

- *다음 명령을 사용하여 **Stratis** 풀을 시작합니다. **--unlock-method** 옵션은 암호화된 경우 풀 잠금 해제 방법을 지정합니다.*

```
# stratis pool start pool-uuid --unlock-method <keyring|clevis>
```

- *또는 다음 명령을 사용하여 **Stratis** 풀을 중지합니다. 이렇게 하면 스토리지 스택이 줄어들지만 모든 메타데이터는 그대로 유지됩니다.*

```
# stratis pool stop pool-name
```

검증 단계

- *다음 명령을 사용하여 시스템의 모든 풀을 나열합니다.*

```
# stratis pool list
```

- *다음 명령을 사용하여 이전에 시작한 풀을 모두 나열합니다. **UUID**를 지정하면 명령은 **UUID**에 해당하는 풀에 대한 세부 정보를 출력합니다.*

```
# stratis pool list --stopped --uuid UUID
```

39.14. STRATIS 파일 시스템 생성

*기존 **Stratis** 풀에 **Stratis** 파일 시스템을 생성합니다.*

사전 요구 사항

- ***Stratis**가 설치되어 있습니다. 자세한 내용은 **Stratis** 설치를 참조하십시오.*

- ***stratisd** 서비스가 실행 중입니다.*

- ***Stratis** 풀을 생성했습니다. 암호화되지 않은 **Stratis** 풀 생성 또는 암호화된 **Stratis** 풀 생성을 참조하십시오.*

절차

1. *풀에서 **Stratis** 파일 시스템을 생성하려면 다음을 사용합니다.*

```
# stratis filesystem create --size number-and-unit my-pool my-fs
```

다음과 같습니다.

number-and-unit

*파일 시스템의 크기를 지정합니다. 사양 형식은 입력의 표준 크기 사양 형식(**B, KiB, MiB, GiB, TiB** 또는 **PiB**)을 따라야 합니다.*

my-pool

***Stratis** 풀의 이름을 지정합니다.*

my-fs

파일 시스템에 대한 임의의 이름을 지정합니다.

예를 들면 다음과 같습니다.

*예 **39.1. Stratis** 파일 시스템 생성*

```
# stratis filesystem create --size 10GiB pool1 filesystem1
```

검증 단계

- *풀을 사용하여 파일 시스템을 나열하여 **Stratis** 파일 시스템이 생성되었는지 확인합니다.*

```
# stratis fs list my-pool
```

추가 리소스

- Stratis 파일 시스템 마운트.

39.15. STRATIS 파일 시스템 마운트

기존 Stratis 파일 시스템을 마운트하여 콘텐츠에 액세스합니다.

사전 요구 사항

- Stratis가 설치되어 있습니다. 자세한 내용은 Stratis 설치를 참조하십시오.
- stratisd 서비스가 실행 중입니다.
- Stratis 파일 시스템을 생성했습니다. 자세한 내용은 Stratis 파일 시스템 생성을 참조하십시오.

절차

- 파일 시스템을 마운트하려면 Stratis가 /dev/stratis/ 디렉터리에서 유지보수하는 항목을 사용합니다.

```
# mount /dev/stratis/my-pool/my-fs mount-point
```

이제 파일 시스템이 마운트 지점 디렉터리에 마운트되어 사용할 준비가 되었습니다.

추가 리소스

- Stratis 파일 시스템 생성.

39.16. STRATIS 파일 시스템 영구적으로 마운트

이 절차에서는 시스템을 부팅한 후 자동으로 사용할 수 있도록 Stratis 파일 시스템을 영구적으로 마운트합니다.

사전 요구 사항

- **Stratis**가 설치되어 있습니다. **Stratis** 설치를 참조하십시오.
- **stratisd** 서비스가 실행 중입니다.
- **Stratis** 파일 시스템을 생성했습니다. **Stratis** 파일 시스템 생성을 참조하십시오.

절차

1. 파일 시스템의 **UUID** 속성을 확인합니다.

```
$ lsblk --output=UUID /dev/stratis/my-pool/my-fs
```

예를 들면 다음과 같습니다.

예 39.2. Stratis 파일 시스템의 UUID 보기

```
$ lsblk --output=UUID /dev/stratis/my-pool/fs1

UUID
a1f0b64a-4ebb-4d4e-9543-b1d79f600283
```

2. 마운트 지점 디렉터리가 없으면 생성합니다.

```
# mkdir --parents mount-point
```

3. **root**로 **/etc/fstab** 파일을 편집하고 **UUID**로 식별되는 파일 시스템의 행을 추가합니다. 파일 시스템 유형으로 **xfs** 를 사용하고 **x-systemd.requires=stratisd.service** 옵션을 추가합니다.

예를 들면 다음과 같습니다.

예 39.3. /etc/fstab의 /fs1 마운트 지점

```
UUID=a1f0b64a-4ebb-4d4e-9543-b1d79f600283 /fs1 xfs defaults,x-
systemd.requires=stratisd.service 0 0
```

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

```
# systemctl daemon-reload
```

5. 파일 시스템을 마운트하여 구성이 작동하는지 확인합니다.

```
# mount mount-point
```

추가 리소스

- 파일 시스템을 영구적으로 마운트합니다.

39.17. SYSTEMD 서비스를 사용하여 /ETC/FSTAB에 루트가 아닌 STRATIS 파일 시스템 설정

systemd 서비스를 사용하여 **/etc/fstab**에서 루트가 아닌 파일 시스템을 설정할 수 있습니다.

사전 요구 사항

- **Stratis**가 설치되어 있습니다. **Stratis** 설치를 참조하십시오.
- **stratisd** 서비스가 실행 중입니다.
- **Stratis** 파일 시스템을 생성했습니다. **Stratis** 파일 시스템 생성을 참조하십시오.

절차

- 루트가 아닌 **Stratis** 파일 시스템에 대해 다음을 사용하십시오.

```
# /dev/stratis/[STRATIS_SYMLINK] [MOUNT_POINT] xfs defaults, x-
systemd.requires=stratis-fstab-setup@[POOL_UUID].service,x-systemd.after=stratis-stab-
setup@[POOL_UUID].service <dump_value> <fsck_value>
```

추가 리소스

- 파일 시스템을 영구적으로 마운트합니다.

40장. 추가 블록 장치를 사용하여 STRATIS 볼륨 확장

Stratis 풀에 추가 블록 장치를 연결하여 Stratis 파일 시스템에 추가 스토리지 용량을 제공할 수 있습니다.

중요

Stratis는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. Red Hat은 프로덕션 환경에서 사용하지 않는 것이 좋습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다. Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 https://access.redhat.com/support/offerings/techpreview 에서 참조하십시오.

40.1. STRATIS 볼륨의 구성 요소

Stratis 볼륨을 구성하는 구성 요소에 대해 알아봅니다.

외부에서 Stratis는 명령줄 인터페이스 및 API에 다음 볼륨 구성 요소를 제공합니다.

blockdev

디스크 또는 디스크 파티션과 같은 블록 장치.

pool

하나 이상의 블록 장치로 구성됩니다.

풀은 블록 장치의 크기와 같은 고정된 전체 크기를 갖습니다.

이 풀에는 dm-cache 대상을 사용하는 비휘발성 데이터 캐시와 같은 대부분의 Stratis 계층이 포함되어 있습니다.

Stratis는 각 풀에 대해 /dev/stratis/my-pool/ 디렉터리를 생성합니다. 이 디렉터리에는 풀의 Stratis 파일 시스템을 나타내는 장치에 대한 링크가 포함되어 있습니다.

filesystem

각 풀에는 파일을 저장하는 하나 이상의 파일 시스템이 포함될 수 있습니다.

파일 시스템은 씬 프로비저닝되며 전체 크기가 고정되어 있지 않습니다. 파일 시스템의 실제 크기는 저장된 데이터와 함께 증가합니다. 데이터 크기가 파일 시스템의 가상 크기에 도달하면 **Stratis**에서 씬 볼륨과 파일 시스템을 자동으로 확장합니다.

파일 시스템은 **XFS**로 포맷됩니다.

중요

Stratis는 **Stratis**를 사용하여 생성된 파일 시스템에 대한 정보를 추적하며, **XFS**를 사용하여 변경한 내용은 **Stratis**에서 자동으로 업데이트되지 않습니다. **Stratis**에서 관리하는 **XFS** 파일 시스템을 다시 포맷하거나 재구성해서는 안 됩니다.

Stratis는 **/dev/stratis/my-pool/my-fs** 경로에 파일 시스템에 대한 링크를 생성합니다.

참고

Stratis는 **dmsetup** 목록 및 **/proc/partitions** 파일에 표시되는 많은 장치 매퍼 장치를 사용합니다. 마찬가지로 **lsblk** 명령 출력은 **Stratis**의 내부 작업 및 계층을 반영합니다.

40.2. STRATIS 풀에 블록 장치 추가

이 절차에서는 **Stratis** 파일 시스템에서 사용할 수 있도록 하나 이상의 블록 장치를 **Stratis** 풀에 추가합니다.

사전 요구 사항

- **Stratis**가 설치되어 있습니다. **Stratis 설치**를 참조하십시오.
- **stratisd** 서비스가 실행 중입니다.
- **Stratis** 풀에 추가하는 블록 장치는 사용되지 않으며 마운트되지 않습니다.

- **Stratis** 풀에 추가하는 블록 장치는 크기가 각각 **1GiB** 이상입니다.

절차

- 풀에 블록 장치를 하나 이상 추가하려면 다음을 사용합니다.

```
# stratis pool add-data my-pool device-1 device-2 device-n
```

추가 리소스

- **Stratis(8)** 도움말 페이지

40.3. 추가 리소스

- **Stratis** 스토리지 웹 사이트

41장. STRATIS 파일 시스템 모니터링

Stratis 사용자는 시스템의 Stratis 볼륨에 대한 정보를 보고 해당 상태 및 여유 공간을 모니터링할 수 있습니다.

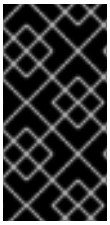
중요

Stratis는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. Red Hat은 프로덕션 환경에서 사용하지 않는 것이 좋습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다. Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 https://access.redhat.com/support/offerings/techpreview 에서 참조하십시오.

41.1. 다양한 유틸리티에서 보고한 STRATIS 크기

이 섹션에서는 df 및 stratis 유틸리티와 같은 표준 유틸리티에서 보고한 Stratis 크기 간의 차이점을 설명합니다.

df 와 같은 표준 Linux 유틸리티는 Stratis의 XFS 파일 시스템 계층(예: 1TiB)을 보고합니다. Stratis의 실제 스토리지 사용량이 씬 프로비저닝으로 인해 덜하기 때문에 유용한 정보가 아니며, XFS 계층이 가까이 있을 때 Stratis가 파일 시스템을 자동으로 늘리기 때문입니다.



중요

Stratis 파일 시스템에 기록된 데이터의 양을 정기적으로 모니터링합니다. 이는 Total Physical Used 값으로 보고됩니다. 합계 물리적 크기 값을 초과하지 않는지 확인합니다.

추가 리소스

- Stratis(8) 도움말 페이지.

41.2. STRATIS 볼륨에 대한 정보 표시

이 절차에서는 Stratis 볼륨에 대한 통계(예: 풀 소유의 총, 사용된 총 크기, 여유 크기 또는 파일 시스템, 블록 장치)를 나열합니다.

사전 요구 사항

- **Stratis**가 설치되어 있습니다. **Stratis** 설치를 참조하십시오.
- **stratisd** 서비스가 실행 중입니다.

절차

- 시스템에서 **Stratis**에 사용되는 모든 블록 장치에 대한 정보를 표시하려면 다음을 수행하십시오.

```
# stratis blockdev

Pool Name  Device Node    Physical Size   State  Tier
my-pool    /dev/sdb            9.10 TiB  In-use  Data
```

- 시스템의 모든 **Stratis** 풀에 대한 정보를 표시하려면 다음을 수행합니다.

```
# stratis pool

Name    Total Physical Size  Total Physical Used
my-pool           9.10 TiB             598 MiB
```

- 시스템의 모든 **Stratis** 파일 시스템에 대한 정보를 표시하려면 다음을 수행합니다.

```
# stratis filesystem

Pool Name  Name  Used     Created           Device
my-pool    my-fs 546 MiB  Nov 08 2018 08:03  /dev/stratis/my-pool/my-fs
```

추가 리소스

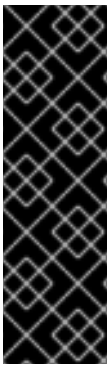
- **Stratis(8)** 도움말 페이지.

41.3. 추가 리소스

- **Stratis** 스토리지 웹 사이트.

42장. STRATIS 파일 시스템에서 스냅샷 사용

Stratis 파일 시스템의 스냅샷을 사용하여 임의의 시간에 파일 시스템 상태를 캡처하고 나중에 복원할 수 있습니다.



중요

Stratis는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. Red Hat은 프로덕션 환경에서 사용하지 않는 것이 좋습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다. Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 https://access.redhat.com/support/offerings/techpreview 에서 참조하십시오.

42.1. STRATIS 스냅샷의 특성

Stratis에서 스냅샷은 다른 Stratis 파일 시스템의 사본으로 생성된 일반 Stratis 파일 시스템입니다. 스냅샷에는 처음에 원본 파일 시스템과 동일한 파일 콘텐츠가 포함되어 있지만 스냅샷이 수정됨에 따라 변경될 수 있습니다. 스냅샷에 변경한 내용은 원본 파일 시스템에 반영되지 않습니다.

Stratis의 현재 스냅샷 구현은 다음과 같은 특징이 있습니다.

- 파일 시스템의 스냅샷은 또 다른 파일 시스템입니다.
- 스냅샷과 원본은 수명으로 연결되지 않습니다. 스냅샷된 파일 시스템은 생성된 파일 시스템보다 오래 보관할 수 있습니다.
- 파일 시스템에서 스냅샷을 만들기 위해 파일 시스템을 마운트할 필요가 없습니다.
- 각 스냅샷은 XFS 로그에 필요한 약 절반의 기가바이트의 실제 백업 스토리지를 사용합니다.

42.2. STRATIS 스냅샷 생성

이 절차에서는 기존 Stratis 파일 시스템의 스냅샷으로 Stratis 파일 시스템을 생성합니다.

사전 요구 사항

- **Stratis**가 설치되어 있습니다. **Stratis** 설치를 참조하십시오.
- **stratisd** 서비스가 실행 중입니다.
- **Stratis** 파일 시스템을 생성했습니다. **Stratis** 파일 시스템 생성을 참조하십시오.

절차

- **Stratis** 스냅샷을 생성하려면 다음을 사용합니다.

```
# stratis fs snapshot my-pool my-fs my-fs-snapshot
```

추가 리소스

- **Stratis(8)** 도움말 페이지.

42.3. STRATIS 스냅샷의 콘텐츠에 액세스

이 절차에서는 **Stratis** 파일 시스템의 스냅샷을 마운트하여 읽기 및 쓰기 작업을 위해 액세스할 수 있도록 합니다.

사전 요구 사항

- **Stratis**가 설치되어 있습니다. **Stratis** 설치를 참조하십시오.
- **stratisd** 서비스가 실행 중입니다.
- **Stratis** 스냅샷을 생성하셨습니다. **Stratis** 파일 시스템 생성을 참조하십시오.

절차

- 스냅샷에 액세스하려면 **/dev/stratis/my-pool/** 디렉토리에서 일반 파일 시스템으로 마운트합니다.

```
# mount /dev/stratis/my-pool/my-fs-snapshot mount-point
```

추가 리소스

- Stratis 파일 시스템 마운트.
- mount(8) 도움말 페이지.

42.4. STRATIS 파일 시스템을 이전 스냅샷으로 되돌리기

이 절차에서는 Stratis 파일 시스템의 콘텐츠를 Stratis 스냅샷에서 캡처된 상태로 되돌립니다.

사전 요구 사항

- Stratis가 설치되어 있습니다. Stratis 설치를 참조하십시오.
- stratisd 서비스가 실행 중입니다.
- Stratis 스냅샷을 생성하셨습니다. Stratis 스냅샷 생성을 참조하십시오.

절차

1. 선택적으로 파일 시스템의 현재 상태를 백업하여 나중에 액세스할 수 있습니다.

```
# stratis filesystem snapshot my-pool my-fs my-fs-backup
```

2. 원래 파일 시스템을 마운트 해제하고 제거합니다.

```
# umount /dev/stratis/my-pool/my-fs
# stratis filesystem destroy my-pool my-fs
```

3. 원본 파일 시스템의 이름으로 스냅샷 사본을 생성합니다.

```
# stratis filesystem snapshot my-pool my-fs-snapshot my-fs
```

4. 원래 파일 시스템과 동일한 이름으로 현재 액세스할 수 있는 스냅샷을 마운트합니다.

```
# mount /dev/stratis/my-pool/my-fs mount-point
```

my-fs라는 파일 시스템의 내용이 **my-fs -snapshot** 스냅샷과 동일합니다.

추가 리소스

- **Stratis(8)** 도움말 페이지.

42.5. STRATIS 스냅샷 제거

이 절차에서는 풀에서 **Stratis** 스냅샷을 제거합니다. 스냅샷의 데이터가 손실됩니다.

사전 요구 사항

- **Stratis**가 설치되어 있습니다. **Stratis** 설치를 참조하십시오.
- **stratisd** 서비스가 실행 중입니다.
- **Stratis** 스냅샷을 생성하셨습니다. **Stratis** 스냅샷 생성을 참조하십시오.

절차

1. 스냅샷을 마운트 해제합니다.

```
# umount /dev/stratis/my-pool/my-fs-snapshot
```

2. 스냅샷을 삭제합니다.

```
# stratis filesystem destroy my-pool my-fs-snapshot
```

추가 리소스

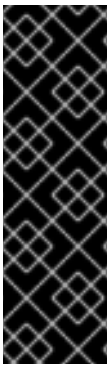
- *Stratis(8) 도움말 페이지.*

42.6. 추가 리소스

- *Stratis 스토리지 웹 사이트.*

43장. STRATIS 파일 시스템 제거

기존 Stratis 파일 시스템 또는 Stratis 풀을 제거하여 데이터를 삭제할 수 있습니다.



중요

Stratis는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. Red Hat은 프로덕션 환경에서 사용하지 않는 것이 좋습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다. Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 https://access.redhat.com/support/offerings/techpreview 에서 참조하십시오.

43.1. STRATIS 볼륨의 구성 요소

Stratis 볼륨을 구성하는 구성 요소에 대해 알아봅니다.

외부에서 Stratis는 명령줄 인터페이스 및 API에 다음 볼륨 구성 요소를 제공합니다.

blockdev

디스크 또는 디스크 파티션과 같은 블록 장치.

pool

하나 이상의 블록 장치로 구성됩니다.

풀은 블록 장치의 크기와 같은 고정된 전체 크기를 갖습니다.

이 풀에는 dm-cache 대상을 사용하는 비휘발성 데이터 캐시와 같은 대부분의 Stratis 계층이 포함되어 있습니다.

Stratis는 각 풀에 대해 /dev/stratis/my-pool/ 디렉터리를 생성합니다. 이 디렉터리에는 풀의 Stratis 파일 시스템을 나타내는 장치에 대한 링크가 포함되어 있습니다.

filesystem

각 풀에는 파일을 저장하는 하나 이상의 파일 시스템이 포함될 수 있습니다.

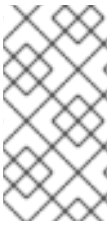
파일 시스템은 씬 프로비저닝되며 전체 크기가 고정되어 있지 않습니다. 파일 시스템의 실제 크기는 저장된 데이터와 함께 증가합니다. 데이터 크기가 파일 시스템의 가상 크기에 도달하면 **Stratis**에서 씬 볼륨과 파일 시스템을 자동으로 확장합니다.

파일 시스템은 **XFS**로 포맷됩니다.

중요

Stratis는 **Stratis**를 사용하여 생성된 파일 시스템에 대한 정보를 추적하며, **XFS**를 사용하여 변경한 내용은 **Stratis**에서 자동으로 업데이트되지 않습니다. **Stratis**에서 관리하는 **XFS** 파일 시스템을 다시 포맷하거나 재구성해서는 안 됩니다.

Stratis는 **/dev/stratis/my-pool/my-fs** 경로에 파일 시스템에 대한 링크를 생성합니다.



참고

Stratis는 **dmsetup** 목록 및 **/proc/partitions** 파일에 표시되는 많은 장치 매퍼 장치를 사용합니다. 마찬가지로 **lsblk** 명령 출력은 **Stratis**의 내부 작업 및 계층을 반영합니다.

43.2. STRATIS 파일 시스템 제거

이 절차에서는 기존 **Stratis** 파일 시스템을 제거합니다. 저장된 데이터는 손실됩니다.

사전 요구 사항

- **Stratis**가 설치되어 있습니다. **Stratis 설치**를 참조하십시오.
- **stratisd** 서비스가 실행 중입니다.
- **Stratis** 파일 시스템을 생성했습니다. **Stratis 파일 시스템 생성**을 참조하십시오.

절차

1. 파일 시스템을 마운트 해제합니다.

```
# umount /dev/stratis/my-pool/my-fs
```

2. 파일 시스템을 삭제합니다.

```
# stratis filesystem destroy my-pool my-fs
```

3. 파일 시스템이 더 이상 존재하지 않는지 확인합니다.

```
# stratis filesystem list my-pool
```

추가 리소스

- **Stratis(8)** 도움말 페이지.

43.3. STRATIS 풀 제거

이 절차에서는 기존 **Stratis** 풀을 제거합니다. 저장된 데이터는 손실됩니다.

사전 요구 사항

- **Stratis**가 설치되어 있습니다. **Stratis** 설치를 참조하십시오.
- **stratisd** 서비스가 실행 중입니다.
- **Stratis** 풀을 생성했습니다.
  - 암호화되지 않은 풀을 생성하려면 암호화되지 않은 **Stratis** 풀 생성을참조하십시오.
  - 암호화된 풀을 생성하려면 암호화된 **Stratis** 풀 생성을 참조하십시오.

절차

1. 풀의 파일 시스템을 나열합니다.

```
# stratis filesystem list my-pool
```

2. 풀에서 모든 파일 시스템을 마운트 해제합니다.

```
# umount /dev/stratis/my-pool/my-fs-1 \
        /dev/stratis/my-pool/my-fs-2 \
        /dev/stratis/my-pool/my-fs-n
```

3. 파일 시스템을 삭제합니다.

```
# stratis filesystem destroy my-pool my-fs-1 my-fs-2
```

4. 풀을 삭제합니다.

```
# stratis pool destroy my-pool
```

5. 풀이 더 이상 존재하지 않는지 확인합니다.

```
# stratis pool list
```

추가 리소스

- **Stratis(8)** 도움말 페이지.

43.4. 추가 리소스

- **Stratis** 스토리지 웹 사이트.

44장. EXT4 파일 시스템 시작하기

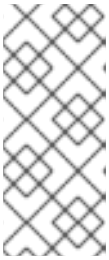
시스템 관리자는 **ext4** 파일 시스템을 생성, 마운트, 크기 조정, 백업 및 복원할 수 있습니다. **ext4** 파일 시스템은 **ext3** 파일 시스템의 확장 가능한 확장입니다. **Red Hat Enterprise Linux 9**를 사용하면 최대 **16 TB**의 개별 파일 크기와 파일 시스템을 최대 **50 TB**까지 지원할 수 있습니다.

44.1. EXT4 파일 시스템의 기능

다음은 **ext4** 파일 시스템의 기능입니다.

- 확장 영역 사용: **ext4** 파일 시스템은 확장 영역을 사용하므로 대용량 파일을 사용할 때 성능이 향상되고 대용량 파일의 메타데이터 오버헤드가 줄어듭니다.
- **Ext4**는 할당되지 않은 블록 그룹 및 **inode** 테이블 섹션에 레이블을 지정하여 파일 시스템 검사 중에 블록 그룹 및 테이블 섹션을 건너뛸 수 있습니다. 이로 인해 파일 시스템의 크기가 증가함에 따라 빠른 파일 시스템 점검이 향상됩니다.
- 메타데이터 체크섬: 이 기능은 **Red Hat Enterprise Linux 9**에서 기본적으로 활성화되어 있습니다.
- **ext4** 파일 시스템의 할당 기능:
  - 영구 사전 할당
  - 지연된 할당
  - 다중 블록 할당
  - 스트라이프 인식 할당
- 확장 속성(**xattr**) 이를 통해 시스템은 파일마다 여러 개의 추가 이름 및 값 쌍을 연결할 수 있습니다.

- 할당량 저널링: 이렇게 하면 충돌 후 긴 할당량 일관성 점검이 필요하지 않습니다.



참고

ext4에서 지원되는 유일한 저널링 모드는 data=ordered (기본값)입니다. 자세한 내용은 RHEL에서 EXT 저널링 옵션 "data=writeback"이 지원되는지참조하십시오. 지식 베이스 문서.

- 하위 시간 타임스탬프 - 이 명령은 타임스탬프를 서브초에 제공합니다.

추가 리소스

- ext4 도움말 페이지.

44.2. EXT4 파일 시스템 만들기

시스템 관리자는 mkfs.ext4 명령을 사용하여 블록 장치에 ext4 파일 시스템을 만들 수 있습니다.

사전 요구 사항

- 디스크의 파티션. GPT 또는 GPT 파티션을 만드는 방법에 대한 자세한 내용은 parted . .로 디스크에서 파티션 테이블 만들기를 참조하십시오.

  또는 LVM 또는 MD 볼륨을 사용하십시오.

절차

1. ext4 파일 시스템을 생성하려면 다음을 수행합니다.

   - 일반 파티션 장치, LVM 볼륨, MD 볼륨 또는 유사한 장치의 경우 다음 명령을 사용합니다.

     ```
     # mkfs.ext4 /dev/block_device
     ```

     /dev/block_device 를 블록 장치의 경로로 바꿉니다.

예를 들어 **/dev/sdb1,/dev/disk/by-uuid/05e99ec8-def1-4a5e-8a9d-59453ceb2a, /dev/my-volgroup/my-lv**. 일반적으로 기본 옵션은 대부분의 사용 시나리오에 적합합니다.

- 스트라이핑된 블록 장치(예: **RAID5** 배열)의 경우 파일 시스템 생성 시 스트라이프 지오메트리를 지정할 수 있습니다. 적절한 스트라이프 지오메트리를 사용하면 **ext4** 파일 시스템의 성능이 향상됩니다. 예를 들어 **4k** 블록 파일 시스템에서 **64k stride (즉, 16 x 4096)**를 가진 파일 시스템을 만들려면 다음 명령을 사용하십시오.

```
# mkfs.ext4 -E stride=16,stripe-width=64 /dev/block_device
```

주어진 예에서:

  - **stride=value: RAID** 청크 크기를 지정합니다.
  - **stripe-width=value: RAID** 장치의 데이터 디스크 수 또는 스트라이프의 스트라이프 단위 수를 지정합니다.

참고

- 파일 시스템을 생성할 때 **UUID**를 지정하려면 다음을 수행합니다.

```
# mkfs.ext4 -U UUID /dev/block_device
```

UUID 를 설정할 **UUID**로 바꿉니다(예: **7cd65de3-e0be-41d9-b66d-96d749c02da7**).

UUID가 추가되도록 **/dev/block_device** 를 **ext4** 파일 시스템의 경로로 바꿉니다(예: **/dev/sda8**).

- 파일 시스템을 생성할 때 라벨을 지정하려면 다음을 수행합니다.

```
# mkfs.ext4 -L label-name /dev/block_device
```

2. 생성된 **ext4** 파일 시스템을 보려면 다음을 수행합니다.

```
# blkid
```

추가 리소스

- **ext4** 도움말 페이지.
- **mkfs.ext4** 도움말 페이지.

44.3. EXT4 파일 시스템 마운트

시스템 관리자는 **mount** 유틸리티를 사용하여 **ext4** 파일 시스템을 마운트 할 수 있습니다.

사전 요구 사항

- **ext4** 파일 시스템. **ext4** 파일 시스템 생성에 대한 자세한 내용은 **ext4 파일 시스템 생성**을 참조하십시오.

절차

1. 파일 시스템을 마운트할 마운트 지점을 생성하려면 다음을 수행합니다.

```
# mkdir /mount/point
```

/mount/point 를 파티션의 마운트 지점을 생성해야 하는 디렉토리 이름으로 바꿉니다.

2. **ext4** 파일 시스템을 마운트하려면 다음을 수행합니다.

- 추가 옵션 없이 **ext4** 파일 시스템을 마운트하려면 다음을 수행합니다.

```
# mount /dev/block_device /mount/point
```

- 파일 시스템을 영구적으로 마운트하려면 영구적으로 **파일 시스템 마운트** 를 참조하십시오.

3. 마운트된 파일 시스템을 보려면 다음을 수행합니다.

```
# df -h
```

추가 리소스

- 도움말 페이지 마운트.
- **ext4** 도움말 페이지.
- **fstab** 도움말 페이지.
- 파일 시스템 마운트 .

44.4. EXT4 파일 시스템 크기 조정

시스템 관리자는 **resize2fs** 유틸리티를 사용하여 **ext4** 파일 시스템의 크기를 조정할 수 있습니다. **resize2fs** 유틸리티는 특정 단위를 나타내는 접미사가 사용되지 않는 한 파일 시스템 블록 크기 단위로 크기를 읽습니다. 다음 접미사는 특정 단위를 나타냅니다.

- **s (sectors) - 512** 바이트 섹터
- **K (KB) - 1,024** 바이트
- **m(메가바이트) - 1,048,576** 바이트
- **G (gigabytes) - 1,073,741,824** 바이트
- **t (terabytes) - 1,099,511,627,776** 바이트

사전 요구 사항

-

ext4 파일 시스템. ext4 파일 시스템 생성에 대한 자세한 내용은 ext4 파일 시스템 생성을 참조하십시오.

- *크기 조정 후 파일 시스템을 보유하는 적절한 크기의 기본 블록 장치입니다.*

절차

1. *ext4 파일 시스템의 크기를 조정하려면 다음 단계를 수행합니다.*

   - *마운트 해제된 ext4 파일 시스템의 크기를 줄이고 확장하려면 다음을 수행합니다.*

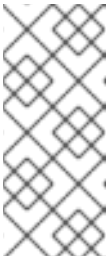
   ```
   # umount /dev/block_device
   # e2fsck -f /dev/block_device
   # resize2fs /dev/block_device size
   ```

   /dev/block_device 를 블록 장치의 경로로 바꿉니다(예: /dev/sdb1).

   s,K,M,G, T 접미사를 사용하여 size 를 필요한 크기 조정 값으로 바꿉니다.

   - *resize2fs 명령을 사용하여 마운트되는 동안 ext4 파일 시스템을 확장할 수 있습니다.*

   ```
   # resize2fs /mount/device size
   ```

   참고

   size 매개변수는 확장할 때 선택 사항(및 중복되는 경우)입니다. resize2fs 는 자동으로 확장하여 사용 가능한 컨테이너 공간(일반적으로 논리 볼륨 또는 파티션)을 채웁니다.

2. *크기가 조정된 파일 시스템을 보려면 다음을 수행합니다.*

   ```
   # df -h
   ```

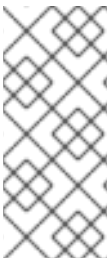
추가 리소스

- *크기 조정**2fs** 도움말 페이지.*
- ***e2fsck** 매뉴얼 페이지.*
- ***ext4** 도움말 페이지.*

44.5. EXT4 및 XFS와 함께 사용되는 툴 비교

*이 섹션에서는 **ext4** 및 **XFS** 파일 시스템에서 일반적인 작업을 수행하는 데 사용할 툴을 비교합니다.*

Task	ext4	XFS
파일 시스템 생성	**mkfs.ext4**	**mkfs.xfs**
파일 시스템 확인	**e2fsck**	**xfs_repair**
파일 시스템 크기 조정	**resize2fs**	**xfs_growfs**
파일 시스템의 이미지 저장	**e2image**	**xfs_metadump** 및 **xfs_mdrestore**
파일 시스템의 레이블 또는 튜닝	**tune2fs**	**xfs_admin**
파일 시스템 백업	**tar** 및 **rsync**	**xfsdump** 및 **xfsrestore**
할당량 관리	**할당량**	**xfs_quota**
파일 매핑	**filefrag**	**xfs_bmap**



참고

*네트워크를 통한 백업에 대한 전체 클라이언트-서버 솔루션을 원한다면 **RHEL 9**에서 사용할 수 있는 **bacula** 백업 유틸리티를 사용할 수 있습니다. **Bacula**에 대한 자세한 내용은 **Bacula 백업 솔루션을** 참조하십시오.*

45장. RHEL 시스템 역할을 사용하여 EXT4 파일 시스템 생성 및 마운트

이 섹션에서는 디스크에 지정된 레이블이 있는 **ext4** 파일 시스템을 생성하고 **storage** 역할을 사용하여 파일 시스템을 영구적으로 마운트하는 방법을 설명합니다.

사전 요구 사항

- 스토리지 역할을 포함한 **Ansible** 플레이북이 있습니다.

45.1. 예시 ANSIBLE PLAYBOOK: EXT4 파일 시스템을 생성 및 마운트

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 **storage** 역할을 적용하여 **Ext4** 파일 시스템을 생성하고 마운트합니다.

예 45.1. /dev/sdb에 Ext4를 생성하고 /mnt/data에 마운트하는 플레이북

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: ext4
        fs_label: label-name
        mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- **Playbook**은 **/dev/sdb** 디스크에 파일 시스템을 생성합니다.
- 플레이북은 **/mnt/data** 디렉터리에 파일 시스템을 영구적으로 마운트합니다.
- 파일 시스템의 레이블은 **label-name** 입니다.

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일.

45.2. 추가 리소스

- 스토리지 역할에 대한 자세한 내용은 다음을 참조하십시오.

- 스토리지 역할 소개