참조
Trident

NetApp
July 01, 2026

목차

참조

Trident 포트

Trident가 통신에 사용하는 포트에 대해 자세히 알아보십시오.

개요

Trident는 Kubernetes 클러스터 내부 및 스토리지 백엔드와의 통신을 위해 다양한 포트를 사용합니다. 다음은 주요 포트, 해당 용도 및 보안 고려 사항에 대한 요약입니다.

- 아웃바운드 포커스: Kubernetes 노드(컨트롤러 및 작업자)는 주로 스토리지 LIF/IP로 트래픽을 시작하므로 iptables 규칙은 노드 IP에서 이러한 포트의 특정 스토리지 IP로의 아웃바운드를 허용해야 합니다. 광범위한 "any-to-any" 규칙을 피하십시오.
- 인바운드 제한: 내부 Trident 포트를 클러스터 내부 트래픽으로 제한합니다(예: Calico와 같은 CNI 사용). 호스트 방화벽에서 불필요한 인바운드 노출이 없습니다.
- 프로토콜 보안:
  - 가능한 경우 TCP를 사용하십시오(더 안정적임).
  - iSCSI의 보안이 중요한 경우 CHAP/IPsec을 활성화하고, 관리에는 TLS/HTTPS(포트 443/8443)를 사용하십시오.
  - NFSv4(Trident의 기본값)의 경우, 필요하지 않다면 UDP/이전 NFSv3 포트(예: 4045-4049)를 제거하십시오.
  - 신뢰할 수 있는 서브넷으로 제한하고 Prometheus와 같은 도구를 사용하여 모니터링하십시오(선택 사항 포트 8001).

컨트롤러 노드용 포트

이 포트는 주로 Trident 운영자(백엔드 관리)를 위한 것입니다. 모든 내부 포트는 Pod 레벨이며, 호스트 방화벽이 CNI를 방해하는 경우에만 노드에서 허용합니다.

포트/프로토콜	방향	목적	드라이버/프로토콜	보안 참고 사항
TCP 8000	인바운드/아웃바운드(클러스터 내부)	Trident REST 서버(operator-controller 통신)	모두	Pod CIDR로 제한하고 외부 노출을 금지합니다.
TCP 8443	인바운드/아웃바운드(클러스터 내부)	백채널 HTTPS(보안 내부 API)	모두	TLS로 암호화됨, Kubernetes 서비스 메시를 사용하는 경우 해당 서비스 메시로 제한됩니다.
TCP 8001	인바운드(클러스터 내부, 선택 사항)	Prometheus 메트릭	모두	모니터링 툴에만 노출(예: RBAC 사용), 사용하지 않을 경우 비활성화하십시오.
TCP 443	아웃바운드	ONTAP SVM/클러스터 관리 LIF에 대한 HTTPS	ONTAP(모두), ANF	TLS 인증서 유효성 검사가 필요하며, 관리 LIF IP로만 제한합니다.
TCP 8443	아웃바운드	E-Series 웹 서비스 프록시에 대한 HTTPS	E-Series(iSCSI)	기본 REST API, 인증서 사용, 백엔드 YAML에서 구성 가능.

작업자 노드용 포트

이 포트는 CSI 노드 데몬셋 및 POD 마운트용입니다. 데이터 포트는 스토리지 데이터 LIF로 아웃바운드되며, NFSv3을 사용하는 경우 NFSv3 추가 항목을 포함합니다(NFSv4의 경우 선택 사항).

포트/프로토콜	방향	목적	드라이버/프로토콜	보안 참고 사항
TCP 17546	인바운드(포드에 대한 로컬)	CSI 노드 활성/준비 프로브	모두	구성 가능(--probe-port), 호스트 충돌이 없는지 확인, 로컬 전용.
TCP 8000	인바운드/아웃바운드(클러스터 내부)	Trident REST 서버	모두	위와 같이, pod 내부입니다.
TCP 8443	인바운드/아웃바운드(클러스터 내부)	백채널 HTTPS	모두	위와 같습니다.
TCP 8001	인바운드(클러스터 내부, 선택 사항)	Prometheus 메트릭	모두	위와 같습니다.
TCP 443	아웃바운드	ONTAP SVM/클러스터 관리 LIF에 대한 HTTPS	ONTAP(모두), ANF	위와 같음, 검색에 사용됩니다.
TCP 8443	아웃바운드	E-Series 웹 서비스 프록시에 대한 HTTPS	E-Series(iSCSI)	위와 같습니다.
TCP/UDP 111	아웃바운드	RPCBIND/portmapper	ONTAP-NAS(NFSv3/v4), ANF(NFS)	v3에 필요하며, v4에서는 선택 사항입니다(방화벽 오프로드). NFSv4 전용을 사용하는 경우 제한하십시오.
TCP/UDP 2049	아웃바운드	NFS 데몬	ONTAP-NAS(NFSv3/v4), ANF(NFS)	핵심 데이터, 잘 알려진 데이터, 안정성을 위해 TCP를 사용합니다.
TCP/UDP 635	아웃바운드	마운트 데몬	ONTAP-NAS(NFSv3/v4), ANF(NFS)	마운팅, 양방향 콜백 가능(필요한 경우 인바운드 임시 허용).
UDP 4045	아웃바운드	NFS 잠금 관리자(nlockmgr)	ONTAP-NAS(NFSv3)	파일 잠금, v4(pNFS 핸들)의 경우 건너뛰기, UDP 전용.
UDP 4046	아웃바운드	NFS 상태 모니터(statd)	ONTAP-NAS(NFSv3)	알림, 콜백을 위해 인바운드 임시 포트(1024-65535)가 필요할 수 있습니다.
UDP 4049	아웃바운드	NFS 할당량 데몬(rquotad)	ONTAP-NAS(NFSv3)	할당량, v4의 경우 건너뜁니다.
TCP 3260	아웃바운드	iSCSI 타겟(검색/데이터/CHAP)	ONTAP-SAN(iSCSI), E-Series(iSCSI)	잘 알려진 기능입니다. 이 포트를 통한 CHAP 인증, 보안을 위해 상호 CHAP를 활성화하십시오.

포트/프로토콜	방향	목적	드라이버/프로토콜	보안 참고 사항
TCP 445	아웃바운드	SMB/CIFS	ONTAP-NAS(SMB), ANF(SMB)	잘 알려진 사실입니다. 암호화와 함께 SMB3 사용(Trident 주석 netapp.io/smb-encryption=true).
TCP/UDP 88(선택 사항)	아웃바운드	Kerberos 인증	ONTAP(Kerb를 사용하는 NFS/SMB/iSCSI)	Kerberos를 사용하는 경우(기본값 아님), 스토리지 시스템이 아닌 AD 서버에 적용됩니다.
TCP/UDP 389(선택 사항)	아웃바운드	LDAP	ONTAP(LDAP를 사용하는 NFS/SMB)	유사, 이름 확인/인증의 경우 AD로 제한합니다.

활성/준비 상태 프로브 포트는 설치 중에 `--probe-port` 플래그를 사용하여 변경할 수 있습니다. 워커 노드에서 다른 프로세스가 이 포트를 사용하고 있지 않은지 확인하는 것이 중요합니다.

Trident REST API

"tridentctl 명령 및 옵션"는 Trident REST API와 상호 작용하는 가장 쉬운 방법이지만, 원한다면 REST 엔드포인트를 직접 사용할 수도 있습니다.

REST API를 사용하는 경우

REST API는 Kubernetes가 아닌 환경에서 Trident를 독립 실행형 바이너리로 사용하는 고급 설치에 유용합니다.

보안 강화를 위해 Trident `REST API`는 `Pod` 내에서 실행될 때 기본적으로 `localhost`로 제한됩니다. 이 동작을 변경하려면 `Pod` 구성에서 `Trident`의 `-address` 인수를 설정해야 합니다.

REST API 사용

이러한 API가 호출되는 방법을 보여주는 예제를 보려면 디버그 (`-d` 플래그를 전달하세요. 자세한 내용은 "tridentctl을 사용하여 Trident를 관리합니다"을 참조하십시오.

API는 다음과 같이 작동합니다.

GET

`GET <trident-address>/trident/v1/<object-type>`

해당 유형의 모든 객체를 나열합니다.

`GET <trident-address>/trident/v1/<object-type>/<object-name>`

명명된 객체의 세부 정보를 가져옵니다.

POST

POST <trident-address>/trident/v1/<object-type>

지정된 유형의 객체를 생성합니다.

- 생성할 오브젝트에 대한 JSON 구성이 필요합니다. 각 오브젝트 유형에 대한 사양은 "tridentctl을 사용하여 Trident를 관리합니다"을 참조하세요.
- 객체가 이미 존재하는 경우 동작이 달라집니다. 백엔드는 기존 객체를 업데이트하지만 다른 모든 객체 유형은 작업에 실패합니다.

삭제

DELETE <trident-address>/trident/v1/<object-type>/<object-name>

명명된 리소스를 삭제합니다.

> 백엔드 또는 스토리지 클래스와 관련된 볼륨은 계속 존재하므로 별도로 삭제해야 합니다. 자세한 내용은 "tridentctl을 사용하여 Trident를 관리합니다"을 참조하십시오.

명령줄 옵션

Trident는 Trident 오케스트레이터에 대한 여러 명령줄 옵션을 제공합니다. 이러한 옵션을 사용하여 배포를 수정할 수 있습니다.

로깅

-debug

디버깅 출력을 활성화합니다.

-loglevel <level>

로깅 수준(debug, info, warn, error, fatal)을 설정합니다. 기본값은 info입니다.

Kubernetes

-k8s_pod

이 옵션 또는 `-k8s_api_server`을 사용하여 Kubernetes 지원을 활성화합니다. 이 옵션을 설정하면 Trident가 포함된 파드의 Kubernetes 서비스 계정 자격 증명을 사용하여 API 서버에 연결합니다. 이는 Trident가 서비스 계정을 활성화한 Kubernetes 클러스터에서 파드로 실행되는 경우에만 작동합니다.

-k8s_api_server <insecure-address:insecure-port>

이 옵션 또는 `-k8s_pod`을 사용하여 Kubernetes 지원을 활성화합니다. 지정하면 Trident는 제공된 비보안 주소와 포트를 사용하여 Kubernetes API 서버에 연결합니다. 이렇게 하면 Trident를 포드 외부에 배포할 수 있지만 API 서버에 대한 비보안 연결만 지원합니다. 안전하게 연결하려면 `-k8s_pod` 옵션을 사용하여 Trident를 포드에 배포합니다.

Docker

-volume_driver <name>

Docker 플러그인을 등록할 때 사용되는 드라이버 이름. 기본값은 `netapp`입니다.

-driver_port <port-number>

UNIX 도메인 소켓이 아닌 이 포트에서 수신 대기합니다.

-config <file>

필수; 백엔드 구성 파일에 이 경로를 지정해야 합니다.

REST

-address <ip-or-host>

Trident의 REST 서버가 수신 대기해야 하는 주소를 지정합니다. 기본값은 localhost입니다. localhost에서 수신 대기하고 Kubernetes 포드 내에서 실행할 때 REST 인터페이스는 포드 외부에서 직접 액세스할 수 없습니다. `-address ""`을(를) 사용하여 포드 IP 주소에서 REST 인터페이스에 액세스할 수 있도록 합니다.

Trident REST 인터페이스는 127.0.0.1(IPv4의 경우) 또는 [::1](IPv6의 경우)에서만 수신 대기하고 서비스를 제공하도록 구성할 수 있습니다.

-port <port-number>

Trident의 REST 서버가 수신 대기해야 하는 포트를 지정합니다. 기본값은 8000입니다.

-rest

REST 인터페이스를 활성화합니다. 기본값은 true입니다.

Kubernetes 및 Trident 객체

REST API를 사용하여 리소스 객체를 읽고 쓰면 Kubernetes 및 Trident와 상호 작용할 수 있습니다. Kubernetes와 Trident, Trident와 스토리지, 그리고 Kubernetes와 스토리지 간의 관계를 정의하는 여러 리소스 객체가 있습니다. 이러한 객체 중 일부는 Kubernetes를 통해 관리되고, 나머지는 Trident를 통해 관리됩니다.

객체는 서로 어떻게 상호 작용합니까?

객체, 그 용도, 그리고 상호 작용 방식을 이해하는 가장 쉬운 방법은 아마도 Kubernetes 사용자의 스토리지 요청 하나를 따라가는 것일 겁니다.

1. 사용자가 관리자가 이전에 구성한 Kubernetes `StorageClass`에서 특정 크기의 새로운 `PersistentVolume`을 요청하는 `PersistentVolumeClaim`을 생성합니다.
2. Kubernetes `StorageClass`는 Trident를 프로비저너로 식별하고 요청된 클래스에 대한 볼륨을 프로비저닝하는 방법을 Trident에 알려주는 매개 변수를 포함합니다.
3. Trident는 해당 클래스에 볼륨을 프로비저닝할 수 있도록 일치하는 `StorageClass`을 식별하는 동일한 이름의 자체 `Backends` 및 `StoragePools`을 확인합니다.
4. Trident는 일치하는 백엔드에 스토리지를 프로비저닝하고 두 개의 객체를 생성합니다. 하나는 Kubernetes에 볼륨을 찾고, 마운트하고, 처리하는 방법을 알려주는 `PersistentVolume`이고, 다른 하나는 `PersistentVolume`와 실제 스토리지 간의 관계를 유지하는 Trident의 볼륨입니다.
5. Kubernetes는 `PersistentVolumeClaim`를 새 `PersistentVolume`에 바인딩합니다. `PersistentVolumeClaim`를 포함하는 Pod는 실행되는 모든 호스트에 해당 PersistentVolume을 마운트합니다.

6. 사용자는 기존 PVC의 `VolumeSnapshot`를 생성하며, Trident를 가리키는 `VolumeSnapshotClass`를 사용합니다.
7. Trident는 PVC와 연결된 볼륨을 식별하고 백엔드에 해당 볼륨의 스냅샷을 생성합니다. 또한 Kubernetes가 스냅샷을 식별하는 방법을 알려주는 `VolumeSnapshotContent`도 생성합니다.
8. 사용자는 `PersistentVolumeClaim`를 생성할 수 있으며, `VolumeSnapshot`를 소스로 사용할 수 있습니다.
9. Trident는 필요한 스냅샷을 식별하고 `PersistentVolume` 및 `Volume`를 생성하는 것과 관련된 동일한 일련의 단계를 수행합니다.

Kubernetes 객체에 대한 자세한 내용은 Kubernetes 문서의 "영구 볼륨" 섹션을 읽어보시기를 강력히 권장합니다.

Kubernetes `PersistentVolumeClaim` 객체

Kubernetes `PersistentVolumeClaim` 객체는 Kubernetes 클러스터 사용자가 스토리지에 대해 요청하는 사항입니다.

표준 사양 외에도 Trident를 사용하면 백엔드 구성에서 설정한 기본값을 재정의하려는 경우 다음과 같은 볼륨별 주석을 지정할 수 있습니다.

주석	볼륨 옵션	지원되는 드라이버
trident.netapp.io/fileSystem	fileSystem	ontap-san, solidfire-san, ontap-san-economy
trident.netapp.io/cloneFromPVC	cloneSourceVolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ontap-nas, ontap-san
trident.netapp.io/protocol	프로토콜	모두
trident.netapp.io/exportPolicy	exportPolicy	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	snapshotPolicy	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotReserve	snapshotReserve	ontap-nas, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotDirectory	snapshotDirectory	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/blockSize	blockSize	SolidFire-SAN
trident.netapp.io/skipRecoveryQueue	skipRecoveryQueue	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy

생성된 PV에 `Delete` 회수 정책이 있는 경우 Trident는 PV가 해제될 때(즉, 사용자가 PVC를 삭제할 때) PV와 백업 볼륨을 모두 삭제합니다. 삭제 작업이 실패하면 Trident는 해당 PV를 표시하고 성공하거나 PV가 수동으로 삭제될

때까지 주기적으로 작업을 재시도합니다. PV가 `Retain` 정책을 사용하는 경우 Trident는 이를 무시하고 관리자가 Kubernetes 및 백엔드에서 정리할 것으로 간주하여 볼륨을 제거하기 전에 백업하거나 검사할 수 있도록 합니다. PV를 삭제해도 Trident가 백업 볼륨을 삭제하지 않습니다. REST API(`tridentctl`를 사용하여 제거해야 합니다.

Trident는 CSI 사양을 사용하여 볼륨 스냅샷 생성을 지원합니다. 볼륨 스냅샷을 생성하고 이를 데이터 소스로 사용하여 기존 PVC를 복제할 수 있습니다. 이렇게 하면 특정 시점의 PV 복사본을 스냅샷 형태로 Kubernetes에 제공할 수 있습니다. 그런 다음 이러한 스냅샷을 사용하여 새로운 PV를 생성할 수 있습니다. `On-Demand Volume Snapshots`작동 방식은 다음을 참조하십시오.

Trident는 `cloneFromPVC` 및 `splitOnClone` 어노테이션도 제공하여 클론을 생성합니다. 이러한 어노테이션을 사용하면 CSI 구현을 사용하지 않고도 PVC를 복제할 수 있습니다.

예를 들어: 사용자가 이미 PVC라는 이름의 `mysql`를 가지고 있다면, 주석을 사용하여 `mysqlclone`라는 새 PVC를 `trident.netapp.io/cloneFromPVC: mysql`와 같이 생성할 수 있습니다. 이 주석이 설정되면, Trident는 새로 볼륨을 프로비저닝하는 대신 mysql PVC에 해당하는 볼륨을 복제합니다.

다음 사항을 고려하십시오.

- NetApp에서는 유휴 볼륨을 복제하는 것을 권장합니다.
- PVC와 그 클론은 동일한 Kubernetes 네임스페이스에 있어야 하며 동일한 스토리지 클래스를 가져야 합니다.
- `ontap-nas` 및 `ontap-san` 드라이버를 사용할 때, PVC 주석 `trident.netapp.io/splitOnClone`을 `trident.netapp.io/cloneFromPVC`와 함께 설정하는 것이 바람직할 수 있습니다. `trident.netapp.io/splitOnClone`이 `true`로 설정되면, Trident는 복제된 볼륨을 상위 볼륨에서 분리하여, 복제된 볼륨의 수명 주기를 상위 볼륨과 완전히 분리하지만 일부 스토리지 효율성을 잃게 됩니다. `trident.netapp.io/splitOnClone`을 설정하지 않거나 `false`로 설정하면, 상위 볼륨과 복제 볼륨 간에 종속성이 생성되어 복제 볼륨이 먼저 삭제되지 않는 한 상위 볼륨을 삭제할 수 없게 되지만, 백엔드에서 공간 사용량이 줄어듭니다. 복제를 분리하는 것이 합리적인 시나리오는 비어 있는 데이터베이스 볼륨을 복제하는 경우로, 이때 볼륨과 그 복제본이 크게 달라질 것으로 예상되어 ONTAP가 제공하는 스토리지 효율성의 이점을 얻지 못하는 경우입니다.

  `sample-input` 디렉토리에는 `Trident`와 함께 사용할 `PVC` 정의의 예가 포함되어 있습니다. `Trident` 볼륨과 관련된 매개변수 및 설정에 대한 전체 설명은 를 참조하십시오.

Kubernetes `PersistentVolume` 객체

Kubernetes `PersistentVolume` 객체는 Kubernetes 클러스터에서 사용할 수 있도록 제공되는 스토리지 영역을 나타냅니다. 이 객체는 해당 객체를 사용하는 Pod와는 독립적인 수명 주기를 가집니다.

Trident는 프로비저닝하는 볼륨을 기반으로 `PersistentVolume` 객체를 생성하고 Kubernetes 클러스터에 자동으로 등록합니다. 사용자가 직접 관리할 필요가 없습니다.

Trident 기반 `StorageClass`을 참조하는 PVC를 생성하면 Trident는 해당 스토리지 클래스를 사용하여 새 볼륨을 프로비저닝하고 해당 볼륨에 대한 새 PV를 등록합니다. 프로비저닝된 볼륨과 해당 PV를 구성할 때 Trident는 다음 규칙을 따릅니다.

- Trident는 Kubernetes용 PV 이름과 스토리지 프로비저닝에 사용하는 내부 이름을 생성합니다. 두 경우 모두 해당 범위 내에서 이름이 고유하도록 보장합니다.
- 볼륨 크기는 PVC에서 요청된 크기와 최대한 일치하지만 플랫폼에 따라 가장 가까운 할당 가능한 수량으로 반올림될 수 있습니다.

Kubernetes `StorageClass` 객체

Kubernetes `StorageClass` 객체는 `PersistentVolumeClaims`에서 이름으로 지정되어 속성 집합을 가진 스토리지를 프로비저닝합니다. 스토리지 클래스 자체는 사용할 프로비저너를 식별하고 프로비저너가 이해할 수 있는 용어로 속성 집합을 정의합니다.

이는 관리자가 생성하고 관리해야 하는 두 가지 기본 객체 중 하나입니다. 다른 하나는 Trident 백엔드 객체입니다.

Trident를 사용하는 Kubernetes `StorageClass` 객체는 다음과 같습니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters: <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

이러한 매개변수는 Trident에 특화되어 있으며 Trident에 해당 클래스에 대한 볼륨을 프로비저닝하는 방법을 알려줍니다.

스토리지 클래스 매개변수는 다음과 같습니다.

속성	유형	필수	설명
속성	map[string]string	아니요	아래 속성 섹션을 참조하십시오
storagePools	map[string]StringList	아니요	내의 스토리지 풀 목록에 대한 백엔드 이름 매핑
additionalStoragePools	map[string]StringList	아니요	내의 스토리지 풀 목록에 대한 백엔드 이름 매핑
excludeStoragePools	map[string]StringList	아니요	내의 스토리지 풀 목록에 대한 백엔드 이름 매핑

스토리지 속성과 해당 속성의 가능한 값은 스토리지 풀 선택 속성과 Kubernetes 속성으로 분류할 수 있습니다.

스토리지 풀 선택 속성

이러한 매개변수는 특정 유형의 볼륨을 프로비저닝하는 데 사용할 Trident 관리 스토리지 풀을 결정합니다.

속성	유형	값	제공	요청	지원 대상:
미디어[1]	문자열	HDD, 하이브리드, SSD	풀에는 이러한 유형의 미디어가 포함되어 있습니다. 하이브리드는 둘 다를 의미합니다	지정된 미디어 유형	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
provisioningType	문자열	씬, 씩	풀은 이 프로비저닝 방식을 지원합니다	프로비저닝 방법이 지정되었습니다	thick: 모든 ONTAP, thin: 모든 ONTAP 및 SolidFire-SAN
backendType	문자열	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	풀은 이러한 유형의 백엔드에 속합니다	지정된 백엔드	모든 드라이버
스냅샷	불	참, 거짓	풀은 스냅샷이 있는 볼륨을 지원합니다	스냅샷이 활성화된 볼륨	ontap-nas, ontap-san, solidfire-san
클론	불	참, 거짓	풀은 볼륨 복제를 지원합니다	클론이 활성화된 볼륨	ontap-nas, ontap-san, solidfire-san
암호화	불	참, 거짓	스토리지 풀은 암호화된 볼륨을 지원합니다.	암호화가 설정된 볼륨	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	int	양의 정수	풀은 이 범위의 IOPS를 보장할 수 있습니다	볼륨에서 이러한 IOPS를 보장합니다	SolidFire-SAN

[1]: ONTAP Select 시스템에서 지원되지 않습니다

대부분의 경우 요청된 값은 프로비저닝에 직접적인 영향을 미칩니다. 예를 들어, 씩 프로비저닝을 요청하면 씩 프로비저닝된 볼륨이 생성됩니다. 그러나 Element 스토리지 풀은 요청된 값이 아닌 제공된 최소 및 최대 IOPS를 사용하여 QoS 값을 설정합니다. 이 경우 요청된 값은 스토리지 풀을 선택하는 데만 사용됩니다.

이상적으로는 `attributes` 단독으로 사용하여 특정 클래스의 요구 사항을 충족하는 데 필요한 스토리지의 특성을 모델링할 수 있습니다. Trident는 사용자가 지정한 `attributes`의 모든 조건에 맞는 스토리지 풀을 자동으로 검색하고 선택합니다.

`attributes`을(를) 사용하여 클래스에 적합한 풀을 자동으로 선택할 수 없는 경우 `storagePools` 및 `additionalStoragePools` 매개 변수를 사용하여 풀을 더욱 세분화하거나 특정 풀 세트를 선택할 수 있습니다.

`storagePools` 매개변수를 사용하여 지정된 `attributes`와 일치하는 풀 집합을 더욱 제한할 수 있습니다. 즉, Trident는 프로비저닝을 위해 `attributes` 및 `storagePools` 매개변수로 식별된 풀의 교집합을 사용합니다. 두 매개변수를 단독으로 사용하거나 함께 사용할 수 있습니다.

`additionalStoragePools` 매개변수를 사용하면 `attributes` 및 `storagePools` 매개변수로 선택된 풀과 관계없이 Trident가 프로비저닝에 사용하는 풀 집합을 확장할 수 있습니다.

`excludeStoragePools` 매개변수를 사용하여 Trident가 프로비저닝에 사용하는 스토리지 풀 집합을 필터링할 수 있습니다. 이 매개변수를 사용하면 일치하는 모든 스토리지 풀이 제거됩니다.

`storagePools` 및 `additionalStoragePools` 매개변수에서 각 항목은 `<backend>:<storagePoolList>` 형식을 취하며, 여기서 `<storagePoolList>`는 지정된 백엔드에 대한 스토리지 풀의 쉼표로 구분된 목록입니다. 예를 들어, `additionalStoragePools`의 값은 `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`과 같을 수 있습니다. 이러한 목록은 백엔드와 목록 값 모두에 대해 정규 표현식 값을 허용합니다. `tridentctl get backend`을 사용하여 백엔드 및 해당 풀 목록을 가져올 수 있습니다.

Kubernetes 속성

이러한 속성은 동적 프로비저닝 중에 Trident가 스토리지 풀/백엔드를 선택하는 데 영향을 미치지 않습니다. 대신, 이러한 속성은 Kubernetes 영구 볼륨에서 지원하는 매개변수를 제공할 뿐입니다. 워커 노드는 파일 시스템 생성 작업을 담당하며 xfsprogs와 같은 파일 시스템 유틸리티가 필요할 수 있습니다.

속성	유형	값	설명	관련 드라이버	**Kubernetes** 버전
fsType	문자열	ext4, ext3, xfs	블록 볼륨의 파일 시스템 유형	solidfire-san, ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy	모두

속성	유형	값	설명	관련 드라이버	**Kubernetes** 버전
allowVolumeExpansion	boolean	참, 거짓	PVC 크기 증가 지원 활성화 또는 비활성화	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, azure-netapp-files	1.11+
volumeBindingMode	문자열	즉시, WaitForFirstConsumer	볼륨 바인딩 및 동적 프로비저닝이 발생하는 시점을 선택하십시오	모두	1.19 - 1.26

- `fsType` 매개변수는 SAN LUN에 사용할 파일 시스템 유형을 제어하는 데 사용됩니다. 또한, Kubernetes는 스토리지 클래스에 `fsType`가 존재하는지 여부를 통해 파일 시스템이 존재함을 나타냅니다. 볼륨 소유권은 `fsGroup` 보안 컨텍스트를 사용하여 제어할 수 있으며, 이는 `fsType`가 설정된 경우에만 가능합니다. "Kubernetes: Pod 또는 Container에 대한 Security Context 구성"를 참조하여 `fsGroup` 컨텍스트를 사용한 볼륨 소유권 설정에 대한 개요를 확인하십시오. Kubernetes는 다음과 같은 경우에만 `fsGroup` 값을 적용합니다.

  - `fsType`이(가) 스토리지 클래스에 설정됩니다.
  - PVC 액세스 모드는 RWO입니다.

  NFS 스토리지 드라이버의 경우 파일 시스템은 이미 NFS 내보내기의 일부로 존재합니다. `fsGroup`을 사용하려면 스토리지 클래스에서 여전히 `fsType`을 지정해야 합니다. `nfs` 또는 null이 아닌 값으로 설정할 수 있습니다.

- 볼륨 확장에 대한 자세한 내용은 "볼륨 확장"를 참조하십시오.
- Trident 설치 프로그램 번들은 `sample-input/storage-class-*.yaml`에서 Trident와 함께 사용할 수 있는 여러 스토리지 클래스 정의 예제를 제공합니다. Kubernetes 스토리지 클래스를 삭제하면 해당 Trident 스토리지 클래스도 함께 삭제됩니다.

Kubernetes `VolumeSnapshotClass` 객체

Kubernetes `VolumeSnapshotClass` 객체는 `StorageClasses`와 유사합니다. 이러한 객체는 여러 스토리지 클래스를 정의하는 데 도움이 되며 볼륨 스냅샷에서 참조되어 스냅샷을 필요한 스냅샷 클래스와 연결합니다. 각 볼륨 스냅샷은 하나의 볼륨 스냅샷 클래스와 연결됩니다.

스냅샷을 생성하려면 관리자가 `VolumeSnapshotClass`를 정의해야 합니다. 볼륨 스냅샷 클래스는 다음 정의를 사용하여 생성됩니다.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

`driver`는 `csi-snapclass` 클래스의 볼륨 스냅샷 요청이 Trident에 의해 처리되도록 Kubernetes에 지정합니다. `deletionPolicy`는 스냅샷을 삭제해야 할 때 수행할 작업을 지정합니다. `deletionPolicy`가 `Delete`로 설정되면 스냅샷이 삭제될 때 볼륨 스냅샷 객체와 스토리지 클러스터의 기본 스냅샷이 모두 제거됩니다. 반대로 `Retain`로 설정하면 `VolumeSnapshotContent`와 물리적 스냅샷이 유지됩니다.

Kubernetes VolumeSnapshot 객체

Kubernetes VolumeSnapshot 객체는 볼륨의 스냅샷 생성을 요청하는 것입니다. PVC가 사용자가 볼륨을 요청하는 것을 나타내는 것처럼, 볼륨 스냅샷은 사용자가 기존 PVC의 스냅샷 생성을 요청하는 것입니다.

볼륨 스냅샷 요청이 들어오면 Trident는 백엔드에서 해당 볼륨의 스냅샷 생성을 자동으로 관리하고 고유한 VolumeSnapshotContent 객체를 생성하여 스냅샷을 노출합니다. 기존 PVC에서 스냅샷을 생성하고 새 PVC를 생성할 때 해당 스냅샷을 DataSource로 사용할 수 있습니다.

VolumeSnapshot의 수명 주기는 소스 PVC와 무관합니다. 스냅샷은 소스 PVC가 삭제된 후에도 유지됩니다. 연결된 스냅샷이 있는 PVC를 삭제할 때 Trident는 해당 PVC의 백업 볼륨을 삭제 중 상태로 표시하지만 완전히 제거하지는 않습니다. 연결된 모든 스냅샷이 삭제되면 해당 볼륨이 제거됩니다.

Kubernetes VolumeSnapshotContent 객체

Kubernetes VolumeSnapshotContent 객체는 이미 프로비저닝된 볼륨에서 생성된 스냅샷을 나타냅니다. 이는 PersistentVolume`와 유사하며 스토리지 클러스터에 프로비저닝된 스냅샷을 의미합니다. `PersistentVolumeClaim 및 PersistentVolume 객체와 마찬가지로 스냅샷이 생성되면 VolumeSnapshotContent 객체는 스냅샷 생성을 요청한 VolumeSnapshot 객체와 일대일 매핑을 유지합니다.

`VolumeSnapshotContent` 객체에는 `snapshotHandle`와 같이 스냅샷을 고유하게 식별하는 세부 정보가 포함되어 있습니다. 이 `snapshotHandle`는 PV 이름과 `VolumeSnapshotContent` 객체 이름의 고유한 조합입니다.

스냅샷 요청이 들어오면 Trident는 백엔드에서 스냅샷을 생성합니다. 스냅샷이 생성되면 Trident는 VolumeSnapshotContent 객체를 구성하여 Kubernetes API에 스냅샷을 노출합니다.

일반적으로 VolumeSnapshotContent 객체를 관리할 필요는 없습니다. 단, Trident 외부에서 생성된 "볼륨 스냅샷 가져오기"를 원하는 경우는 예외입니다.

Kubernetes VolumeGroupSnapshotClass 객체

Kubernetes VolumeGroupSnapshotClass 객체는 `VolumeSnapshotClass`와 유사합니다. 이러한 객체는 여러 스토리지 클래스를 정의하는 데 도움이 되며, 볼륨 그룹 스냅샷에서 참조되어 스냅샷을 필요한 스냅샷 클래스와 연결합니다. 각 볼륨 그룹 스냅샷은 하나의 볼륨 그룹 스냅샷 클래스와 연결됩니다.

`VolumeGroupSnapshotClass`는 스냅샷 그룹을 생성하기 위해 관리자가 정의해야 합니다. 볼륨 그룹 스냅샷 클래스는 다음 정의를 사용하여 생성됩니다.

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

`driver`는 `csi-group-snap-class` 클래스의 볼륨 그룹 스냅샷 요청이 Trident에서 처리되도록 Kubernetes에 지정합니다. `deletionPolicy`는 그룹 스냅샷을 삭제해야 할 때 수행할 작업을 지정합니다. `deletionPolicy`가 `Delete`로 설정되면 스냅샷이 삭제될 때 볼륨 그룹 스냅샷 객체와 스토리지 클러스터의 기본 스냅샷이 모두 제거됩니다. 반대로 `Retain`로 설정하면 `VolumeGroupSnapshotContent`와 물리적 스냅샷이 유지됩니다.

Kubernetes VolumeGroupSnapshot 객체

Kubernetes VolumeGroupSnapshot 객체는 여러 볼륨의 스냅샷을 생성해 달라는 요청입니다. PVC가 사용자가 볼륨에 대해 요청하는 것을 나타내는 것처럼, 볼륨 그룹 스냅샷은 사용자가 기존 PVC의 스냅샷을 생성해 달라는 요청입니다.

볼륨 그룹 스냅샷 요청이 들어오면 Trident는 백엔드에서 해당 볼륨에 대한 그룹 스냅샷 생성을 자동으로 관리하고 고유한 VolumeGroupSnapshotContent 객체를 생성하여 스냅샷을 노출합니다. 기존 PVC에서 스냅샷을 생성하고 새 PVC를 생성할 때 해당 스냅샷을 DataSource로 사용할 수 있습니다.

VolumeGroupSnapshot의 수명 주기는 소스 PVC와 무관합니다. 소스 PVC가 삭제된 후에도 스냅샷은 유지됩니다. 연결된 스냅샷이 있는 PVC를 삭제할 때 Trident는 해당 PVC의 백업 볼륨을 삭제 중 상태로 표시하지만 완전히 제거하지는 않습니다. 연결된 모든 스냅샷이 삭제되면 볼륨 그룹 스냅샷도 제거됩니다.

Kubernetes VolumeGroupSnapshotContent 객체

Kubernetes VolumeGroupSnapshotContent 객체는 이미 프로비저닝된 볼륨에서 생성된 그룹 스냅샷을 나타냅니다. 이는 PersistentVolume`와 유사하며 스토리지 클러스터에 프로비저닝된 스냅샷을 의미합니다. `PersistentVolumeClaim 및 PersistentVolume 객체와 마찬가지로 스냅샷이 생성되면

VolumeSnapshotContent 객체는 스냅샷 생성을 요청한 VolumeSnapshot 객체와 일대일 매핑을 유지합니다.

`VolumeGroupSnapshotContent` 객체에는 스토리지 시스템에 있는 `volumeGroupSnapshotHandle` 및 개별 volumeSnapshotHandles와 같이 스냅샷 그룹을 식별하는 세부 정보가 포함되어 있습니다.

스냅샷 요청이 들어오면 Trident는 백엔드에서 볼륨 그룹 스냅샷을 생성합니다. 볼륨 그룹 스냅샷이 생성되면 Trident는 VolumeGroupSnapshotContent 객체를 구성하여 Kubernetes API에 스냅샷을 노출합니다.

Kubernetes CustomResourceDefinition 객체

Kubernetes 사용자 정의 리소스는 관리자가 정의하고 유사한 객체를 그룹화하는 데 사용되는 Kubernetes API의 엔드포인트입니다. Kubernetes는 객체 모음을 저장하기 위한 사용자 정의 리소스 생성을 지원합니다. 이러한 리소스 정의는 `kubectl get crds`을 실행하여 얻을 수 있습니다.

사용자 정의 리소스 정의(CRD)와 관련 객체 메타데이터는 Kubernetes의 메타데이터 저장소에 저장됩니다. 따라서 Trident를 위한 별도의 저장소가 필요하지 않습니다.

Trident는 CustomResourceDefinition 객체를 사용하여 Trident 백엔드, Trident 스토리지 클래스, Trident 볼륨과 같은 Trident 객체의 ID를 보존합니다. 이러한 객체는 Trident에서 관리됩니다. 또한 CSI 볼륨 스냅샷 프레임워크는 볼륨 스냅샷을 정의하는 데 필요한 일부 CRD를 도입합니다.

CRD는 Kubernetes 구성 요소입니다. 위에 정의된 리소스의 객체는 Trident에 의해 생성됩니다. 간단한 예로, `tridentctl`을 사용하여 백엔드를 생성하면 Kubernetes에서 사용할 수 있는 해당 `tridentbackends` CRD 객체가 생성됩니다.

Trident의 CRD에 대해 유념해야 할 몇 가지 사항은 다음과 같습니다.

- Trident가 설치되면 CRD 세트가 생성되며 다른 리소스 유형과 마찬가지로 사용할 수 있습니다.
- tridentctl uninstall 명령을 사용하여 Trident를 제거하면 Trident Pod가 삭제되지만 생성된 CRD는 정리되지 않습니다. Trident를 완전히 제거하고 처음부터 다시 구성하는 방법을 이해하려면 "Trident 제거"를 참조하십시오.

Trident StorageClass 오브젝트

Trident는 프로비저너 필드에 csi.trident.netapp.io`를 지정하는 Kubernetes `StorageClass 오브젝트에 대해 일치하는 스토리지 클래스를 생성합니다. 스토리지 클래스 이름은 해당 Kubernetes StorageClass 오브젝트의 이름과 일치합니다.

Kubernetes에서는 Trident를 프로비저너로 사용하는 Kubernetes `StorageClass`가 등록될 때 이러한 객체가 자동으로 생성됩니다.

스토리지 클래스는 볼륨에 대한 요구 사항 집합으로 구성됩니다. Trident는 이러한 요구 사항을 각 스토리지 풀에 있는 특성과 일치시킵니다. 일치하면 해당 스토리지 풀은 해당 스토리지 클래스를 사용하여 볼륨을 프로비저닝하기 위한 유효한 대상이 됩니다.

REST API를 사용하여 스토리지 클래스 구성을 생성하면 스토리지 클래스를 직접 정의할 수 있습니다. 하지만 Kubernetes 배포의 경우, 새로운 Kubernetes StorageClass 객체를 등록할 때 스토리지 클래스 구성이 생성될 것으로 예상합니다.

Trident 백엔드 객체

백엔드는 Trident가 볼륨을 프로비저닝하는 스토리지 공급자를 나타냅니다. 단일 Trident 인스턴스는 여러 백엔드를 관리할 수 있습니다.

> 이는 사용자가 직접 생성하고 관리하는 두 가지 객체 유형 중 하나입니다. 다른 하나는 Kubernetes `StorageClass` 객체입니다.

이러한 객체를 구성하는 방법에 대한 자세한 내용은 "백엔드 구성"을 참조하십시오.

Trident `StoragePool` 오브젝트

스토리지 풀은 각 백엔드에서 프로비저닝에 사용할 수 있는 개별 위치를 나타냅니다. ONTAP의 경우 이는 SVM의 애그리게이트에 해당합니다. NetApp HCI/SolidFire의 경우 이는 관리자가 지정한 QoS 대역에 해당합니다. 각 스토리지 풀에는 성능 특성과 데이터 보호 특성을 정의하는 여러 가지 스토리지 속성이 있습니다.

여기 있는 다른 객체와 달리 스토리지 풀 후보는 항상 자동으로 검색되고 관리됩니다.

Trident `Volume` 오브젝트

볼륨은 프로비저닝의 기본 단위로, NFS 공유, iSCSI 및 FC LUN과 같은 백엔드 엔드포인트를 포함합니다. Kubernetes에서 이러한 볼륨은 `PersistentVolumes`에 직접적으로 대응합니다. 볼륨을 생성할 때는 볼륨의 프로비저닝 위치를 결정하는 스토리지 클래스와 크기를 지정해야 합니다.

> - Kubernetes에서는 이러한 객체가 자동으로 관리됩니다. Trident가 프로비저닝한 내용을 확인할 수 있습니다.
> - 스냅샷이 연결된 PV를 삭제하면 해당 Trident 볼륨의 상태가 *삭제 중*으로 업데이트됩니다. Trident 볼륨을 삭제하려면 해당 볼륨의 스냅샷을 제거해야 합니다.

볼륨 구성은 프로비저닝된 볼륨이 가져야 할 속성을 정의합니다.

속성	유형	필수	설명
버전	문자열	아니요	Trident API 버전("1")
이름	문자열	예	생성할 볼륨의 이름
storageClass	문자열	예	볼륨 프로비저닝 시 사용할 스토리지 클래스
크기	문자열	예	프로비저닝할 볼륨의 크기(바이트)
프로토콜	문자열	아니요	사용할 프로토콜 유형, "file" 또는 "block"
internalName	문자열	아니요	스토리지 시스템의 오브젝트 이름, Trident에서 생성됨
cloneSourceVolume	문자열	아니요	ontap (nas, san) & solidfire-*: 복제할 볼륨의 이름

속성	유형	필수	설명
splitOnClone	문자열	아니요	ONTAP(NAS, SAN): 클론을 상위 항목에서 분할합니다
snapshotPolicy	문자열	아니요	ontap-*: 사용할 스냅샷 정책
snapshotReserve	문자열	아니요	ontap-*: 스냅샷용으로 예약된 볼륨의 백분율
exportPolicy	문자열	아니요	ontap-nas*: 사용할 엑스포트 정책
snapshotDirectory	불	아니요	ontap-nas*: 스냅샷 디렉토리가 표시되는지 여부
unixPermissions	문자열	아니요	ontap-nas*: 초기 UNIX 권한
blockSize	문자열	아니요	SolidFire-*: 블록/섹터 크기
fileSystem	문자열	아니요	파일 시스템 유형
skipRecoveryQueue	문자열	아니요	볼륨 삭제 시 스토리지의 복구 큐를 건너뛰고 볼륨을 즉시 삭제합니다.

Trident는 볼륨을 생성할 때 `internalName`를 생성합니다. 이 과정은 두 단계로 구성됩니다. 첫째, 스토리지 접두사(기본 `trident` 또는 백엔드 구성의 접두사)를 볼륨 이름 앞에 추가하여 `<prefix>-<volume-name>` 형식의 이름을 생성합니다. 그런 다음 백엔드에서 허용되지 않는 문자를 대체하여 이름을 정제합니다. ONTAP 백엔드의 경우 하이픈을 밑줄로 대체합니다(따라서 내부 이름은 `<prefix>_<volume-name>`가 됩니다). Element 백엔드의 경우 밑줄을 하이픈으로 대체합니다.

볼륨 구성을 사용하여 REST API를 통해 볼륨을 직접 프로비저닝할 수 있지만, Kubernetes 배포 환경에서는 대부분의 사용자가 표준 Kubernetes `PersistentVolumeClaim` 방식을 사용할 것으로 예상됩니다. Trident는 프로비저닝 프로세스의 일부로 이 볼륨 객체를 자동으로 생성합니다.

Trident `Snapshot` 오브젝트

스냅샷은 볼륨의 시점 복사본으로, 새 볼륨을 프로비저닝하거나 상태를 복원하는 데 사용할 수 있습니다. Kubernetes에서 스냅샷은 `VolumeSnapshotContent` 객체에 직접적으로 대응합니다. 각 스냅샷은 볼륨과 연결되며, 이 볼륨은 스냅샷의 데이터 소스가 됩니다.

각 `Snapshot` 객체는 아래에 나열된 속성을 포함합니다.

속성	유형	필수	설명
버전	문자열	예	Trident API 버전("1")
이름	문자열	예	Trident 스냅샷 객체의 이름
internalName	문자열	예	스토리지 시스템의 Trident 스냅샷 객체 이름

속성	유형	필수	설명
volumeName	문자열	예	스냅샷이 생성되는 영구 볼륨의 이름입니다
volumeInternalName	문자열	예	스토리지 시스템의 연결된 Trident 볼륨 오브젝트 이름

Kubernetes에서는 이러한 객체가 자동으로 관리됩니다. Trident가 프로비저닝한 내용을 확인할 수 있습니다.

Kubernetes `VolumeSnapshot` 객체 요청이 생성되면 Trident는 백업 스토리지 시스템에서 스냅샷 객체를 생성하여 작동합니다. 이 스냅샷 객체의 `internalName`는 접두사 `snapshot-`와 `UID` 객체의 `VolumeSnapshot`를 결합하여 생성됩니다(예: `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` 및 `volumeInternalName`는 백업 볼륨의 세부 정보를 가져와 채워집니다.

Trident `ResourceQuota` 객체

Trident 데몬셋은 `system-node-critical` 우선순위 클래스(Kubernetes에서 사용 가능한 가장 높은 우선순위 클래스)를 사용하여 정상적인 노드 종료 중에 Trident가 볼륨을 식별하고 정리할 수 있도록 하며, 리소스 부하가 높은 클러스터에서 Trident 데몬셋 파드가 우선순위가 낮은 워크로드를 선점할 수 있도록 합니다.

이를 위해 Trident는 `ResourceQuota` 객체를 사용하여 Trident 데몬셋에서 "system-node-critical" 우선순위 클래스가 충족되도록 합니다. 배포 및 데몬셋 생성 전에 Trident는 `ResourceQuota` 객체를 찾고, 찾지 못하면 적용합니다.

기본 리소스 할당량 및 우선순위 클래스를 더 세밀하게 제어해야 하는 경우 `custom.yaml`를 생성하거나 `Helm` 차트를 사용하여 `ResourceQuota` 객체를 구성할 수 있습니다.

다음은 `ResourceQuota` 객체가 Trident 데몬셋의 우선순위를 지정하는 예입니다.

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
    matchExpressions:
      - operator: In
        scopeName: PriorityClass
        values:
          - system-node-critical
```

리소스 할당량에 대한 자세한 내용은 "Kubernetes: 리소스 할당량"을 참조하십시오.

설치 실패 시 정리 작업 `ResourceQuota`

`ResourceQuota` 객체가 생성된 후 설치가 실패하는 드문 경우에는 먼저 xref:{relative_path}../trident-managing-k8s/uninstall-trident.html["제거"]를 시도한 다음 다시 설치하십시오.

그래도 안 되면 수동으로 `ResourceQuota` 개체를 제거하세요.

제거 `ResourceQuota`

리소스 할당을 직접 제어하려면 다음 명령을 사용하여 Trident `ResourceQuota` 객체를 제거할 수 있습니다.

```
kubectl delete quota trident-csi -n trident
```

Pod Security Standards(PSS) 및 Security Context Constraints(SCC)

Kubernetes Pod 보안 표준(PSS) 및 Pod 보안 정책(PSP)은 권한 수준을 정의하고 Pod의 동작을 제한합니다. OpenShift 보안 컨텍스트 제약 조건(SCC)은 OpenShift Kubernetes Engine에 특화된 Pod 제한 사항을 정의합니다. 이러한 사용자 정의를 제공하기 위해 Trident는 설치 중에 특정 권한을 활성화합니다. 다음 섹션에서는 Trident에서 설정한 권한에 대해 자세히 설명합니다.

PSS는 Pod Security Policies(PSP)를 대체합니다. PSP는 Kubernetes v1.21에서 사용이 권장되지 않으며 v1.25에서 제거될 예정입니다. 자세한 내용은 "Kubernetes: 보안"를 참조하십시오.

필수 Kubernetes 보안 컨텍스트 및 관련 필드

권한	설명
권한 있음	CSI를 사용하려면 마운트 포인트가 양방향이어야 하므로 Trident 노드 Pod는 권한 있는 컨테이너를 실행해야 합니다. 자세한 내용은 "Kubernetes: 마운트 전파"을 참조하십시오.
호스트 네트워킹	iSCSI 데몬에 필요합니다. `iscsiadm`는 iSCSI 마운트를 관리하고 호스트 네트워킹을 사용하여 iSCSI 데몬과 통신합니다.
호스트 IPC	NFS는 프로세스 간 통신(IPC)을 사용하여 NFSD와 통신합니다.
호스트 PID	NFS의 경우 `rpc-statd`을(를) 시작해야 합니다. Trident은 NFS 볼륨을 마운트하기 전에 호스트 프로세스를 쿼리하여 `rpc-statd`이(가) 실행 중인지 확인합니다.

권한	설명
기능	SYS_ADMIN 기능은 권한 있는 컨테이너에 대한 기본 기능의 일부로 제공됩니다. 예를 들어 Docker는 권한 있는 컨테이너에 대해 다음과 같은 기능을 설정합니다: CapPrm: 0000003fffffffff CapEff: 0000003fffffffff
Seccomp	Seccomp 프로필은 권한 있는 컨테이너에서 항상 "제한 없음" 상태이므로 Trident에서 활성화할 수 없습니다.
SELinux	OpenShift에서 권한이 있는 컨테이너는 spc_t("Super Privileged Container") 도메인에서 실행되고, 권한이 없는 컨테이너는 container_t 도메인에서 실행됩니다. containerd`에서 `container-selinux`가 설치된 경우 모든 컨테이너가 `spc_t 도메인에서 실행되므로 SELinux가 사실상 비활성화됩니다. 따라서 Trident는 컨테이너에 `seLinuxOptions`를 추가하지 않습니다.
DAC	권한이 있는 컨테이너는 루트로 실행해야 합니다. 권한이 없는 컨테이너는 CSI에 필요한 Unix 소켓에 액세스하기 위해 루트로 실행됩니다.

Pod Security Standards(PSS)

레이블	설명	기본값
pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version	Trident Controller와 노드가 설치 네임스페이스에 포함될 수 있도록 허용합니다. 네임스페이스 레이블은 변경하지 마십시오.	enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>

네임스페이스 레이블을 변경하면 Pod가 스케줄링되지 않거나 "생성 오류: …" 또는 "경고: trident-csi-… " 오류가 발생할 수 있습니다. 이러한 문제가 발생하면 네임스페이스 레이블이 privileged 변경되었는지 확인하십시오. 변경된 경우 Trident를 다시 설치하십시오.

Pod Security Policies(PSP)

필드	설명	기본값
allowPrivilegeEscalation	권한 있는 컨테이너는 권한 상승을 허용해야 합니다.	true
allowedCSIDrivers	Trident는 인라인 CSI 임시 볼륨을 사용하지 않습니다.	비어 있음
allowedCapabilities	권한이 없는 Trident 컨테이너는 기본 세트 이상의 기능을 필요로 하지 않으며 권한이 있는 컨테이너에는 가능한 모든 기능이 부여됩니다.	비어 있음

필드	설명	기본값
allowedFlexVolumes	Trident는 "FlexVolume 드라이버"를 사용하지 않으므로 허용되는 볼륨 목록에 포함되지 않습니다.	비어 있음
allowedHostPaths	Trident 노드 Pod는 노드의 루트 파일 시스템을 마운트하므로 이 목록을 설정해도 아무런 이점이 없습니다.	비어 있음
allowedProcMountTypes	Trident는 어떤 것도 사용하지 않습니다 ProcMountTypes.	비어 있음
allowedUnsafeSysctls	Trident는 안전하지 않은 `sysctls`을 필요로 하지 않습니다.	비어 있음
defaultAddCapabilities	권한 있는 컨테이너에는 별도의 기능을 추가할 필요가 없습니다.	비어 있음
defaultAllowPrivilegeEscalation	권한 상승 허용은 각 Trident Pod에서 처리됩니다.	false
forbiddenSysctls	아니요 sysctls 허용되지 않습니다.	비어 있음
fsGroup	Trident 컨테이너는 루트로 실행됩니다.	RunAsAny
hostIPC	NFS 볼륨을 마운트하려면 호스트 IPC가 와 통신해야 합니다 nfsd	true
hostNetwork	iscsiadm은 iSCSI 데몬과 통신하기 위해 호스트 네트워크가 필요합니다.	true
hostPID	호스트 PID는 노드에서 `rpc-statd`이(가) 실행 중인지 확인하는 데 필요합니다.	true
hostPorts	Trident는 호스트 포트를 사용하지 않습니다.	비어 있음
privileged	Trident 노드 Pod는 볼륨을 마운트하기 위해 권한 있는 컨테이너를 실행해야 합니다.	true
readOnlyRootFilesystem	Trident 노드 Pod는 노드 파일 시스템에 기록해야 합니다.	false
requiredDropCapabilities	Trident 노드 Pod는 권한 있는 컨테이너를 실행하므로 기능을 삭제할 수 없습니다.	none
runAsGroup	Trident 컨테이너는 루트로 실행됩니다.	RunAsAny
runAsUser	Trident 컨테이너는 루트로 실행됩니다.	runAsAny
runtimeClass	Trident는 `RuntimeClasses`을 사용하지 않습니다.	비어 있음

필드	설명	기본값
seLinux	Trident는 현재 컨테이너 런타임과 Kubernetes 배포판이 SELinux를 처리하는 방식에 차이가 있기 때문에 `seLinuxOptions`을 설정하지 않습니다.	비어 있음
supplementalGroups	Trident 컨테이너는 루트로 실행됩니다.	RunAsAny
volumes	Trident Pod에는 이러한 볼륨 플러그인이 필요합니다.	hostPath, projected, emptyDir

보안 컨텍스트 제약 조건(SCC)

라벨	설명	기본값
allowHostDirVolumePlugin	Trident 노드 Pod는 노드의 루트 파일 시스템을 마운트합니다.	true
allowHostIPC	NFS 볼륨을 마운트하려면 호스트 IPC가 `nfsd`와(과) 통신해야 합니다.	true
allowHostNetwork	iscsiadm은 iSCSI 데몬과 통신하기 위해 호스트 네트워크가 필요합니다.	true
allowHostPID	호스트 PID는 노드에서 `rpc-statd`이(가) 실행 중인지 확인하는 데 필요합니다.	true
allowHostPorts	Trident는 호스트 포트를 사용하지 않습니다.	false
allowPrivilegeEscalation	권한 있는 컨테이너는 권한 상승을 허용해야 합니다.	true
allowPrivilegedContainer	Trident 노드 Pod는 볼륨을 마운트하기 위해 권한 있는 컨테이너를 실행해야 합니다.	true
allowedUnsafeSysctls	Trident는 안전하지 않은 `sysctls`을 필요로 하지 않습니다.	none
allowedCapabilities	권한이 없는 Trident 컨테이너는 기본 세트 이상의 기능을 필요로 하지 않으며 권한이 있는 컨테이너에는 가능한 모든 기능이 부여됩니다.	비어 있음
defaultAddCapabilities	권한 있는 컨테이너에는 별도의 기능을 추가할 필요가 없습니다.	비어 있음
fsGroup	Trident 컨테이너는 루트로 실행됩니다.	RunAsAny
groups	이 SCC는 Trident에 특정되며 해당 사용자에게 귀속됩니다.	비어 있음

라벨	설명	기본값
`readOnlyRootFilesystem`	Trident 노드 Pod는 노드 파일 시스템에 기록해야 합니다.	`false`
`requiredDropCapabilities`	Trident 노드 Pod는 권한 있는 컨테이너를 실행하므로 기능을 삭제할 수 없습니다.	`none`
`runAsUser`	Trident 컨테이너는 루트로 실행됩니다.	`RunAsAny`
`seLinuxContext`	Trident는 현재 컨테이너 런타임과 Kubernetes 배포판이 SELinux를 처리하는 방식에 차이가 있기 때문에 `seLinuxOptions`을 설정하지 않습니다.	비어 있음
`seccompProfiles`	권한이 있는 컨테이너는 항상 "Unconfined"로 실행됩니다.	비어 있음
`supplementalGroups`	Trident 컨테이너는 루트로 실행됩니다.	`RunAsAny`
`users`	이 SCC를 Trident 네임스페이스의 Trident 사용자에게 바인딩하기 위한 항목이 하나 제공됩니다.	해당 없음
`volumes`	Trident Pod에는 이러한 볼륨 플러그인이 필요합니다.	`hostPath, downwardAPI, projected, emptyDir`