참조
Trident

NetApp
January 15, 2026

목차

참조

Trident 포트

Trident 통신에 사용하는 포트에 대해 자세히 알아보세요.

Trident 포트

Trident Kubernetes 내부 통신에 다음 포트를 사용합니다.

포트	목적
8443	백채널 HTTPS
8001	Prometheus 메트릭 엔드포인트
8000	Trident REST 서버
17546	Trident daemonset Pod에서 사용하는 활성/준비 프로브 포트

설치 중에 활성/준비 프로브 포트를 변경할 수 있습니다. `--probe-port` 깃발. 이 포트가 워커 노드의 다른 프로세스에 의해 사용되고 있지 않은지 확인하는 것이 중요합니다.

Trident REST API

하는 동안"tridentctl 명령 및 옵션" Trident REST API와 상호 작용하는 가장 쉬운 방법이며, 원하는 경우 REST 엔드포인트를 직접 사용할 수도 있습니다.

REST API를 사용하는 경우

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

더 나은 보안을 위해 Trident `REST API` Pod 내부에서 실행할 경우 기본적으로 localhost로 제한됩니다. 이 동작을 변경하려면 Trident를 설정해야 합니다. `-address` 포드 구성의 인수입니다.

REST API 사용

이러한 API가 호출되는 방법에 대한 예를 보려면 디버그를 전달하세요.(`-d`) 깃발. 자세한 내용은 다음을 참조하세요 ."tridentctl을 사용하여 Trident 관리" .

API는 다음과 같이 작동합니다.

얻다

`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.

우편

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

지정된 유형의 객체를 생성합니다.

- 객체를 생성하려면 JSON 구성이 필요합니다. 각 객체 유형의 사양은 다음을 참조하세요."tridentctl을 사용하여 Trident 관리" .
- 해당 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>

지정된 리소스를 삭제합니다.

> 백엔드 또는 스토리지 클래스와 연결된 볼륨은 계속 존재합니다. 이러한 볼륨은 별도로 삭제해야 합니다. 자세한 내용은 다음을 참조하세요."tridentctl을 사용하여 Trident 관리" .

명령줄 옵션

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

벌채 반출

-debug

디버깅 출력을 활성화합니다.

-loglevel <level>

로깅 수준(디버그, 정보, 경고, 오류, 치명적)을 설정합니다. 기본적으로 info로 설정됩니다.

쿠버네티스

-k8s_pod

이 옵션을 사용하거나 `-k8s_api_server` Kubernetes 지원을 활성화합니다. 이것을 설정하면 Trident 포함된 Pod의 Kubernetes 서비스 계정 자격 증명을 사용하여 API 서버에 접속합니다. 이 기능은 Trident 서비스 계정이 활성화된 Kubernetes 클러스터에서 Pod로 실행되는 경우에만 작동합니다.

-k8s_api_server <insecure-address:insecure-port>

이 옵션을 사용하거나 `-k8s_pod` Kubernetes 지원을 활성화합니다. 이 옵션을 지정하면 Trident 제공된 안전하지 않은 주소와 포트를 사용하여 Kubernetes API 서버에 연결합니다. 이를 통해 Trident 포드 외부에 배포할 수 있습니다. 하지만 API 서버에 대한 안전하지 않은 연결만 지원합니다. 안전하게 연결하려면 포드에 Trident 배치하세요. `-k8s_pod` 옵션.

도커

-volume_driver <name>

Docker 플러그인을 등록할 때 사용되는 드라이버 이름입니다. 기본값으로 설정됨 `netapp` .

-driver_port <port-number>

UNIX 도메인 소켓이 아닌 이 포트에서 수신합니다.

-config <file>

필수. 백엔드 구성 파일에 대한 경로를 지정해야 합니다.

나머지

-address <ip-or-host>

Trident의 REST 서버가 수신해야 하는 주소를 지정합니다. 기본값은 localhost입니다. 로컬호스트에서 수신하고 Kubernetes Pod 내부에서 실행하는 경우 REST 인터페이스는 Pod 외부에서 직접 액세스할 수 없습니다. 사용 `-address ""` Pod IP 주소에서 REST 인터페이스에 접근할 수 있도록 합니다.

> Trident REST 인터페이스는 127.0.0.1(IPv4의 경우) 또는 [::1](IPv6의 경우)에서만 수신하고 서비스하도록 구성할 수 있습니다.

-port <port-number>

Trident의 REST 서버가 수신해야 하는 포트를 지정합니다. 기본값은 8000입니다.

-rest

REST 인터페이스를 활성화합니다. 기본값은 true입니다.

Kubernetes 및 Trident 객체

REST API를 사용하여 리소스 객체를 읽고 쓰는 방식으로 Kubernetes 및 Trident 와 상호 작용할 수 있습니다. Kubernetes와 Trident, Trident 와 스토리지, Kubernetes와 스토리지 간의 관계를 결정하는 여러 리소스 객체가 있습니다. 이러한 객체 중 일부는 Kubernetes를 통해 관리되고 나머지는 Trident 통해 관리됩니다.

각 객체는 서로 어떻게 상호 작용하나요?

아마도 객체를 이해하고, 객체의 용도와 상호 작용을 파악하는 가장 쉬운 방법은 Kubernetes 사용자의 단일 저장소 요청을 따르는 것입니다.

1. 사용자가 생성합니다 `PersistentVolumeClaim` 새로운 것을 요청하다 `PersistentVolume` Kubernetes에서 특정 크기의 `StorageClass` 이는 이전에 관리자가 구성한 것입니다.
2. 쿠버네티스 `StorageClass` Trident 프로비저너로 식별하고 요청된 클래스에 대한 볼륨을 프로비저닝하는 방법을 Trident 알려주는 매개변수를 포함합니다.
3. Trident 자체적으로 `StorageClass` 일치하는 것을 식별하는 동일한 이름으로 `Backends` 그리고 `StoragePools` 이를 사용하여 클래스에 대한 볼륨을 프로비저닝할 수 있습니다.
4. Trident 일치하는 백엔드에 저장소를 프로비저닝하고 두 개의 객체를 생성합니다. `PersistentVolume`

Kubernetes에서 볼륨을 찾고 마운트하고 처리하는 방법을 Kubernetes에 알려주는 볼륨과 볼륨 간의 관계를 유지하는 Trident 의 볼륨 `PersistentVolume` 그리고 실제 저장 공간.

5. Kubernetes는 다음을 바인딩합니다. `PersistentVolumeClaim` 새로운 것에 `PersistentVolume` . 다음을 포함하는 포드 `PersistentVolumeClaim` 해당 PersistentVolume을 실행 중인 모든 호스트에 마운트합니다.
6. 사용자가 생성합니다 `VolumeSnapshot` 기존 PVC를 사용하여 `VolumeSnapshotClass` Trident 를 가리키죠.
7. Trident PVC와 연관된 볼륨을 식별하고 백엔드에 볼륨의 스냅샷을 만듭니다. 또한 다음을 생성합니다. `VolumeSnapshotContent` Kubernetes에 스냅샷을 식별하는 방법을 지시합니다.
8. 사용자는 다음을 생성할 수 있습니다. `PersistentVolumeClaim` 사용 중 `VolumeSnapshot` 출처로서.
9. Trident 필요한 스냅샷을 식별하고 스냅샷 생성에 관련된 동일한 단계 세트를 수행합니다. `PersistentVolume` 그리고 `Volume` .

Kubernetes 객체에 대한 추가 정보를 읽으려면 다음을 꼭 읽어 보시기 바랍니다. "영구 볼륨" Kubernetes 문서의 섹션입니다.

쿠버네티스 `PersistentVolumeClaim` 사물

쿠버네티스 `PersistentVolumeClaim` 객체는 Kubernetes 클러스터 사용자가 만든 저장소 요청입니다.

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

주석	볼륨 옵션	지원되는 드라이버
트라이던트.넷앱.io/파일시스템	파일 시스템	온탭산, 솔리드파이어산, 온탭산이코노미
트라이던트.넷앱.io/cloneFromPVC	cloneSourceVolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, gcp-cvs, ontap-san-economy
트라이던트.넷앱.io/splitOnClone	splitOnClone	온탑나스, 온탑산
트라이던트.넷앱.io/프로토콜	규약	어느
트라이던트.넷앱.io/내보내기정책	수출정책	온탑나스, 온탑나스이코노미, 온탑나스플렉스그룹
트라이던트.넷앱.io/스냅샷정책	스냅샷 정책	온탑나스, 온탑나스이코노미, 온탑나스플렉스그룹, 온탑산
트라이던트.넷앱.io/스냅샷예약	스냅샷예약	온탭-나스, 온탭-나스-플렉스그룹, 온탭-산, GCP-CVS
trident.netapp.io/스냅샷 디렉토리	스냅샷 디렉토리	온탑나스, 온탑나스이코노미, 온탑나스플렉스그룹
트라이던트.넷앱.io/유닉스권한	unixPermissions	온탑나스, 온탑나스이코노미, 온탑나스플렉스그룹
트라이던트.넷앱.io/블록사이즈	블록 크기	솔리드파이어-산

생성된 PV에 다음이 있는 경우 `Delete` 회수 정책에 따라, Trident PV가 해제되면(즉, 사용자가 PVC를 삭제하면) PV와 백업 볼륨을 모두 삭제합니다. 삭제 작업이 실패하면 Trident PV를 삭제로 표시하고 성공하거나 PV를 수동으로 삭제할 때까지 주기적으로 작업을 다시 시도합니다. PV가 사용하는 경우 `Retain` 정책에 따라 Trident 이를 무시하고

관리자가 Kubernetes와 백엔드에서 이를 정리하여 볼륨을 제거하기 전에 백업하거나 검사할 수 있다고 가정합니다. PV를 삭제해도 Trident 백업 볼륨을 삭제하지는 않습니다. REST API를 사용하여 제거해야 합니다.(`tridentctl`).

Trident CSI 사양을 사용하여 볼륨 스냅샷 생성을 지원합니다. 볼륨 스냅샷을 생성하고 이를 데이터 소스로 사용하여 기존 PVC를 복제할 수 있습니다. 이런 방식으로 PV의 특정 시점 복사본을 스냅샷 형태로 Kubernetes에 노출할 수 있습니다. 그런 다음 스냅샷을 사용하여 새로운 PV를 만들 수 있습니다. 살펴보세요 `On-Demand Volume Snapshots` 이것이 어떻게 작동하는지 알아보려고요.

Trident 또한 다음을 제공합니다. `cloneFromPVC` 그리고 `splitOnClone` 클론을 생성하기 위한 주석. 이러한 주석을 사용하면 CSI 구현을 사용하지 않고도 PVC를 복제할 수 있습니다.

다음은 예입니다. 사용자가 이미 PVC를 가지고 있는 경우 `mysql` 사용자는 새로운 PVC를 생성할 수 있습니다. `mysqlclone` 예를 들어 주석을 사용하여 `trident.netapp.io/cloneFromPVC: mysql` . 이 주석 세트를 사용하면 Trident 볼륨을 처음부터 프로비저닝하는 대신 MySQL PVC에 해당하는 볼륨을 복제합니다.

다음 사항을 고려하세요.

- NetApp 유휴 볼륨을 복제할 것을 권장합니다.
- PVC와 해당 복제본은 동일한 Kubernetes 네임스페이스에 있어야 하며 동일한 스토리지 클래스를 가져야 합니다.
- 와 함께 `ontap-nas` 그리고 `ontap-san` 드라이버의 경우 PVC 주석을 설정하는 것이 바람직할 수 있습니다. `trident.netapp.io/splitOnClone` 와 함께 `trident.netapp.io/cloneFromPVC` . 와 함께 `trident.netapp.io/splitOnClone` 로 설정 `true` Trident 복제된 볼륨을 부모 볼륨에서 분리하여 복제된 볼륨의 수명 주기를 부모 볼륨에서 완전히 분리하지만, 저장 효율성은 일부 떨어집니다. 설정하지 않음 `trident.netapp.io/splitOnClone` 또는 그것을 설정 `false` 부모 볼륨과 복제 볼륨 간에 종속성이 생성되어 복제 볼륨을 먼저 삭제하지 않으면 부모 볼륨을 삭제할 수 없게 되면서 백엔드에서 공간 사용량이 줄어듭니다. 복제본을 분할하는 것이 합리적인 시나리오 중 하나는 볼륨과 복제본이 크게 갈라지고 ONTAP 이 제공하는 저장 효율성의 이점을 얻지 못할 것으로 예상되는 빈 데이터베이스 볼륨을 복제하는 것입니다.

그만큼 `sample-input` 디렉토리에는 Trident 와 함께 사용할 수 있는 PVC 정의의 예가 포함되어 있습니다. 참조하다 Trident 볼륨과 관련된 매개변수와 설정에 대한 전체 설명을 확인하세요.

쿠버네티스 `PersistentVolume` 사물

쿠버네티스 `PersistentVolume` 객체는 Kubernetes 클러스터에서 사용할 수 있는 저장소를 나타냅니다. 포드를 사용하는 포드와 독립적인 수명 주기를 갖습니다.

Trident 생성 `PersistentVolume` 객체를 생성하고 프로비저닝하는 볼륨에 따라 자동으로 Kubernetes 클러스터에 등록합니다. 당신이 직접 관리할 필요는 없습니다.

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

- Trident Kubernetes에 대한 PV 이름과 스토리지를 프로비저닝하는 데 사용하는 내부 이름을 생성합니다. 두 경우 모두 이름이 해당 범위에서 고유하다는 점이 보장됩니다.
- 볼륨 크기는 PVC에서 요청한 크기와 최대한 일치하도록 하지만, 플랫폼에 따라 가장 가까운 할당 가능한 수량으로 반올림될 수 있습니다.

쿠버네티스 StorageClass 사물

쿠버네티스 StorageClass 객체는 이름으로 지정됩니다. PersistentVolumeClaims 속성 집합으로 저장소를 프로비저닝합니다. 스토리지 클래스 자체는 사용될 프로비저너를 식별하고 프로비저너가 이해하는 용어로 해당 속성 집합을 정의합니다.

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

쿠버네티스 StorageClass Trident 사용하는 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 Trident에만 적용되며 Trident 클래스에 대한 볼륨을 프로비저닝하는 방법을 알려줍니다.

저장 클래스 매개변수는 다음과 같습니다.

기인하다	유형	필수의	설명
속성	맵[문자열]문자열	아니요	아래의 속성 섹션을 참조하세요.
스토리지풀	map[문자열]문자열목록	아니요	백엔드 이름 맵을 스토리지 풀 목록으로
추가 스토리지 풀	map[문자열]문자열목록	아니요	백엔드 이름 맵을 스토리지 풀 목록으로
스토리지 풀 제외	map[문자열]문자열목록	아니요	백엔드 이름 맵을 스토리지 풀 목록으로

저장소 속성과 가능한 값은 저장소 풀 선택 속성과 Kubernetes 속성으로 분류할 수 있습니다.

스토리지 풀 선택 속성

이러한 매개변수는 주어진 유형의 볼륨을 프로비저닝하는 데 어떤 Trident 관리 스토리지 풀을 사용해야 하는지 결정합니다.

기인하다	유형	가치	권하다	요구	지원됨
미디어[1]	끈	HDD, 하이브리드, SSD	풀에는 이 유형의 미디어가 포함되어 있습니다. 하이브리드는 둘 다 의미합니다.	지정된 미디어 유형	온탑-나스, 온탑-나스-이코노미, 온탑-나스-플렉스그룹, 온탑-산, 솔리드파이어-산
프로비저닝 유형	끈	얇은, 두꺼운	풀은 이 프로비저닝 방법을 지원합니다.	프로비저닝 방법이 지정됨	두꺼운: 모두 온탭; 얇은: 모두 온탭 & solidfire-san
백엔드 유형	끈	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, gcp-cvs, azure-netapp-files, ontap-san-economy	풀은 이 유형의 백엔드에 속합니다.	백엔드 지정됨	모든 운전자
스냅샷	부울	참, 거짓	풀은 스냅샷이 있는 볼륨을 지원합니다.	스냅샷이 활성화된 볼륨	온탭-나스, 온탭-산, 솔리드파이어-산, GCP-CVS
클론	부울	참, 거짓	풀은 볼륨 복제를 지원합니다.	복제가 활성화된 볼륨	온탭-나스, 온탭-산, 솔리드파이어-산, GCP-CVS
암호화	부울	참, 거짓	풀은 암호화된 볼륨을 지원합니다.	암호화가 활성화된 볼륨	온탑나스, 온탑나스이코노미, 온탑나스플렉스그룹, 온탑산
아이옵스	정수	양의 정수	풀은 이 범위에서 IOPS를 보장할 수 있습니다.	볼륨은 이러한 IOPS를 보장합니다.	솔리드파이어-산

[1]: ONTAP Select 시스템에서는 지원되지 않습니다.

대부분의 경우, 요청된 값은 프로비저닝에 직접적인 영향을 미칩니다. 예를 들어, 두꺼운 프로비저닝을 요청하면 두꺼운 프로비저닝된 볼륨이 생성됩니다. 하지만 Element 스토리지 풀은 요청된 값이 아닌 제공된 IOPS 최소값과 최대값을 사용하여 QoS 값을 설정합니다. 이 경우, 요청된 값은 스토리지 풀을 선택하는 데에만 사용됩니다.

이상적으로는 다음을 사용할 수 있습니다. `attributes` 특정 계층의 요구를 충족하는 데 필요한 저장소의 품질을 단독으로 모델링합니다. Trident 모든 것과 일치하는 스토리지 풀을 자동으로 검색하고 선택합니다. `attributes` 당신이 지정한 대로.

사용할 수 없는 경우 `attributes` 수업에 적합한 풀을 자동으로 선택하려면 다음을 사용할 수 있습니다. `storagePools` 그리고 `additionalStoragePools` 풀을 더욱 세분화하거나 특정 풀 세트를 선택하기 위한 매개변수입니다.

당신은 사용할 수 있습니다 `storagePools` 지정된 풀과 일치하는 풀 세트를 추가로 제한하기 위한 매개변수

`attributes` . 즉, Trident 다음에 의해 식별된 풀의 교차점을 사용합니다. `attributes` 그리고 `storagePools` 프로비저닝을 위한 매개변수. 두 매개변수 중 하나만 사용하거나 두 매개변수를 함께 사용할 수 있습니다.

당신은 사용할 수 있습니다 `additionalStoragePools` Trident 프로비저닝에 사용하는 풀 세트를 확장하기 위한 매개변수입니다. 이는 선택한 풀에 관계없이 적용됩니다. `attributes` 그리고 `storagePools` 매개변수.

당신은 사용할 수 있습니다 `excludeStoragePools` Trident 프로비저닝에 사용하는 풀 세트를 필터링하는 매개변수입니다. 이 매개변수를 사용하면 일치하는 모든 풀이 제거됩니다.

에서 `storagePools` 그리고 `additionalStoragePools` 매개변수, 각 항목은 다음 형식을 취합니다. `<backend>:<storagePoolList>` , 어디 `<storagePoolList>` 지정된 백엔드에 대한 스토리지 풀의 쉼표로 구분된 목록입니다. 예를 들어, 값 `additionalStoragePools` 처럼 보일 수도 있습니다 `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze` . 이러한 목록은 백엔드 값과 목록 값 모두에 대한 정규식 값을 허용합니다. 사용할 수 있습니다 `tridentctl get backend` 백엔드와 해당 풀의 목록을 가져옵니다.

쿠버네티스 속성

이러한 속성은 동적 프로비저닝 중 Trident 가 스토리지 풀/백엔드를 선택하는 데 영향을 미치지 않습니다. 대신 이러한 속성은 Kubernetes 영구 볼륨에서 지원하는 매개변수를 제공합니다. 워커 노드는 파일 시스템 생성 작업을 담당하며 xfsprogs와 같은 파일 시스템 유틸리티가 필요할 수 있습니다.

기인하다	유형	가치	설명	관련 드라이버	쿠버네티스 버전
fs타입	끈	ext4, ext3, xfs	블록 볼륨의 파일 시스템 유형	솔리드파이어-샌, 온탑-나스, 온탑-나스-이코노미, 온탑-나스-플렉스그룹, 온탑-샌, 온탑-샌-이코노미	모두
볼륨 확장 허용	부울	참, 거짓	PVC 크기 확장 지원 활성화 또는 비활성화	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, gcp-cvs, azure-netapp-files	1.11+
볼륨 바인딩 모드	끈	즉시, WaitForFirstConsumer	볼륨 바인딩 및 동적 프로비저닝이 발생하는 시점을 선택하세요	모두	1.19 - 1.26

- 그만큼 `fsType` 매개변수는 SAN LUN에 대한 원하는 파일 시스템 유형을 제어하는 데 사용됩니다. 또한 Kubernetes는 다음의 존재를 사용합니다. `fsType` 파일 시스템이 존재한다는 것을 나타내기 위해 저장 클래스에 있습니다. 볼륨 소유권은 다음을 사용하여 제어할 수 있습니다. `fsGroup` 포드의 보안 컨텍스트는 다음과 같은 경우에만 해당됩니다. `fsType` 설정되었습니다. 참조하다"Kubernetes: Pod 또는 컨테이너에 대한 보안 컨텍스트 구성" 볼륨 소유권 설정에 대한 개요는 다음을 참조하세요. `fsGroup` 문맥. Kubernetes는 다음을 적용합니다. `fsGroup` 값은 다음의 경우에만 적용됩니다.

  - `fsType`저장 클래스에 설정됩니다.
  - PVC 접근 모드는 RWO입니다.

  NFS 스토리지 드라이버의 경우 파일 시스템은 이미 NFS 내보내기의 일부로 존재합니다. 사용하기 위해 `fsGroup` 저장 클래스는 여전히 다음을 지정해야 합니다. `fsType` . 설정할 수 있습니다 `nfs` 또는 null이 아닌 값.

- 참조하다"볼륨 확장" 볼륨 확장에 대한 자세한 내용은 다음을 참조하세요.
- Trident 설치 프로그램 번들은 Trident 와 함께 사용할 수 있는 여러 가지 예시 스토리지 클래스 정의를 제공합니다.`sample-input/storage-class-*.yaml` . Kubernetes 스토리지 클래스를 삭제하면 해당 Trident 스토리지 클래스도 삭제됩니다.

쿠버네티스 `VolumeSnapshotClass` 사물

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

에이 `VolumeSnapshotClass` 스냅샷을 생성하려면 관리자가 정의해야 합니다. 다음 정의를 사용하여 볼륨 스냅샷 클래스가 생성됩니다.

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

그만큼 `driver` Kubernetes에 볼륨 스냅샷에 대한 요청을 지정합니다. `csi-snapclass` 클래스는 Trident 에 의해 처리됩니다. 그만큼 `deletionPolicy` 스냅샷을 삭제해야 할 때 수행할 작업을 지정합니다. 언제 `deletionPolicy` 로 설정됩니다 `Delete` 스냅샷이 삭제되면 볼륨 스냅샷 개체와 스토리지 클러스터의 기본 스냅샷도 제거됩니다. 또는 다음과 같이 설정합니다. `Retain` 즉, `VolumeSnapshotContent` 그리고 물리적 스냅샷은 보존됩니다.

쿠버네티스 `VolumeSnapshot` 사물

쿠버네티스 `VolumeSnapshot` 객체는 볼륨의 스냅샷을 생성하라는 요청입니다. PVC가 사용자가 볼륨에 대해 한 요청을 나타내는 것처럼 볼륨 스냅샷은 사용자가 기존 PVC의 스냅샷을 생성해 달라는 요청입니다.

볼륨 스냅샷 요청이 들어오면 Trident 백엔드에서 볼륨에 대한 스냅샷 생성을 자동으로 관리하고 고유한 스냅샷을 생성하여 스냅샷을 노출합니다.

VolumeSnapshotContent 물체. 기존 PVC에서 스냅샷을 생성하고 새 PVC를 생성할 때 해당 스냅샷을 DataSource로 사용할 수 있습니다.

> VolumeSnapshot의 수명 주기는 소스 PVC와 무관합니다. 즉, 소스 PVC가 삭제된 후에도 스냅샷은 유지됩니다. 연관된 스냅샷이 있는 PVC를 삭제할 때 Trident 이 PVC의 백업 볼륨을 삭제 상태로 표시하지만 완전히 제거하지는 않습니다. 연관된 모든 스냅샷이 삭제되면 볼륨이 제거됩니다.

쿠버네티스 VolumeSnapshotContent 사물

쿠버네티스 VolumeSnapshotContent 객체는 이미 프로비저닝된 볼륨에서 가져온 스냅샷을 나타냅니다. 이것은 다음과 유사합니다. PersistentVolume 스토리지 클러스터에 프로비저닝된 스냅샷을 나타냅니다. 와 유사하다 PersistentVolumeClaim 그리고 PersistentVolume 스냅샷이 생성되면 개체 VolumeSnapshotContent 객체는 일대일 매핑을 유지합니다. VolumeSnapshot 스냅샷 생성을 요청한 객체입니다.

그만큼 VolumeSnapshotContent 개체에는 스냅샷을 고유하게 식별하는 세부 정보(예: snapshotHandle . 이것 snapshotHandle PV 이름과 VolumeSnapshotContent 물체.

스냅샷 요청이 들어오면 Trident 백엔드에서 스냅샷을 생성합니다. 스냅샷이 생성된 후 Trident 다음을 구성합니다. VolumeSnapshotContent 객체를 생성하여 Kubernetes API에 스냅샷을 노출합니다.

> 일반적으로 관리할 필요가 없습니다. VolumeSnapshotContent 물체. 이에 대한 예외는 다음을 원할 때입니다."볼륨 스냅샷 가져오기" Trident 외부에서 생성됨.

쿠버네티스 VolumeGroupSnapshotClass 사물

쿠버네티스 VolumeGroupSnapshotClass 객체는 다음과 유사합니다 VolumeSnapshotClass . 이러한 스냅샷은 여러 스토리지 클래스를 정의하는 데 도움이 되며 볼륨 그룹 스냅샷에서 참조되어 스냅샷을 필요한 스냅샷 클래스와 연결합니다. 각 볼륨 그룹 스냅샷은 단일 볼륨 그룹 스냅샷 클래스와 연결됩니다.

에이 VolumeGroupSnapshotClass 스냅샷 그룹을 생성하려면 관리자가 정의해야 합니다. 다음 정의를 사용하여 볼륨 그룹 스냅샷 클래스가 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
```

그만큼 driver 볼륨 그룹 스냅샷에 대한 요청을 Kubernetes에 지정합니다. csi-group-snap-class 클래스는 Trident 에 의해 처리됩니다. 그만큼 deletionPolicy 그룹 스냅샷을 삭제해야 할 때 수행할 작업을 지정합니다. 언제 deletionPolicy 로 설정됩니다 Delete 스냅샷이 삭제되면 볼륨 그룹 스냅샷 개체와 스토리지 클러스터의 기본 스냅샷도 제거됩니다. 또는 다음과 같이 설정합니다. Retain 즉, VolumeGroupSnapshotContent 그리고 물리적 스냅샷은 보존됩니다.

쿠버네티스 `VolumeGroupSnapshot` 사물

쿠버네티스 `VolumeGroupSnapshot` 객체는 여러 볼륨의 스냅샷을 생성하라는 요청입니다. PVC가 사용자가 볼륨에 대해 한 요청을 나타내는 것처럼 볼륨 그룹 스냅샷은 사용자가 기존 PVC의 스냅샷을 생성해 달라는 요청입니다.

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

> VolumeGroupSnapshot의 수명 주기는 소스 PVC와 무관합니다. 즉, 소스 PVC가 삭제된 후에도 스냅샷은 유지됩니다. 연관된 스냅샷이 있는 PVC를 삭제할 때 Trident 이 PVC의 백업 볼륨을 삭제 상태로 표시하지만 완전히 제거하지는 않습니다. 연관된 모든 스냅샷이 삭제되면 볼륨 그룹 스냅샷도 제거됩니다.

쿠버네티스 `VolumeGroupSnapshotContent` 사물

쿠버네티스 `VolumeGroupSnapshotContent` 개체는 이미 프로비저닝된 볼륨에서 가져온 그룹 스냅샷을 나타냅니다. 이것은 다음과 유사합니다. `PersistentVolume` 스토리지 클러스터에 프로비저닝된 스냅샷을 나타냅니다. 와 유사하다 `PersistentVolumeClaim` 그리고 `PersistentVolume` 스냅샷이 생성되면 개체 `VolumeSnapshotContent` 객체는 일대일 매핑을 유지합니다. `VolumeSnapshot` 스냅샷 생성을 요청한 객체입니다.

그만큼 `VolumeGroupSnapshotContent` 개체에는 스냅샷 그룹을 식별하는 세부 정보(예: `volumeGroupSnapshotHandle` 그리고 개별 볼륨 스냅샷 핸들이 스토리지 시스템에 존재합니다.

스냅샷 요청이 들어오면 Trident 백엔드에 볼륨 그룹 스냅샷을 생성합니다. 볼륨 그룹 스냅샷이 생성된 후 Trident 다음을 구성합니다. `VolumeGroupSnapshotContent` 객체를 생성하여 Kubernetes API에 스냅샷을 노출합니다.

쿠버네티스 `CustomResourceDefinition` 사물

Kubernetes 사용자 정의 리소스는 관리자가 정의한 Kubernetes API의 엔드포인트이며 유사한 객체를 그룹화하는 데 사용됩니다. Kubernetes는 객체 컬렉션을 저장하기 위한 사용자 정의 리소스 생성을 지원합니다. 다음 리소스 정의는 다음을 실행하여 얻을 수 있습니다. `kubectl get crds` .

사용자 정의 리소스 정의(CRD)와 관련 개체 메타데이터는 Kubernetes의 메타데이터 저장소에 저장됩니다. 이로 인해 Trident 위한 별도의 매장이 필요 없게 되었습니다.

Trident 사용 `CustomResourceDefinition` Trident 백엔드, Trident 스토리지 클래스, Trident 볼륨 등 Trident 객체의 ID를 보존하는 객체입니다. 이러한 객체는 Trident 에서 관리합니다. 또한 CSI 볼륨 스냅샷 프레임워크는 볼륨 스냅샷을 정의하는 데 필요한 일부 CRD를 도입합니다.

CRD는 Kubernetes 구성 요소입니다. 위에 정의된 리소스의 객체는 Trident 에 의해 생성됩니다. 간단한 예로, 백엔드가 생성될 때 `tridentctl` , 해당 `tridentbackends` CRD 객체는 Kubernetes에서 사용하기 위해 생성됩니다.

Trident의 CRD에 대해 염두에 두어야 할 몇 가지 사항은 다음과 같습니다.

- Trident 설치하면 CRD 세트가 생성되어 다른 리소스 유형과 마찬가지로 사용할 수 있습니다.
- 다음을 사용하여 Trident 제거할 때 `tridentctl uninstall` 명령을 실행하면 Trident 포드가 삭제되지만 생성된 CRD는 정리되지 않습니다. 참조하다"Trident 제거" Trident 완전히 제거하고 처음부터 재구성하는 방법을 알아보세요.

Trident `StorageClass` 사물

Trident Kubernetes에 맞는 스토리지 클래스를 생성합니다. `StorageClass` 지정하는 객체 `csi.trident.netapp.io` 해당 프로비저닝 필드에 있습니다. 스토리지 클래스 이름은 Kubernetes의 이름과 일치합니다. `StorageClass` 그것이 나타내는 대상.

> Kubernetes를 사용하면 이러한 객체는 Kubernetes가 자동으로 생성됩니다. `StorageClass` Trident 프로비저너로 사용하는 것이 등록되었습니다.

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

REST API를 사용하여 스토리지 클래스를 직접 정의하기 위해 스토리지 클래스 구성을 생성할 수 있습니다. 그러나 Kubernetes 배포의 경우 새 Kubernetes를 등록할 때 생성될 것으로 예상합니다. `StorageClass` 사물.

Trident 백엔드 객체

백엔드는 Trident 가 볼륨을 프로비저닝하는 스토리지 공급자를 나타냅니다. 단일 Trident 인스턴스는 아무리 많은 백엔드라도 관리할 수 있습니다.

> 이는 사용자가 직접 만들고 관리하는 두 가지 객체 유형 중 하나입니다. 다른 하나는 쿠버네티스입니다 `StorageClass` 물체.

이러한 객체를 구성하는 방법에 대한 자세한 내용은 다음을 참조하세요."백엔드 구성" .

Trident `StoragePool` 사물

스토리지 풀은 각 백엔드에서 프로비저닝에 사용할 수 있는 고유한 위치를 나타냅니다. ONTAP 의 경우 이는 SVM의 집계에 해당합니다. NetApp HCI/ SolidFire 의 경우 이는 관리자가 지정한 QoS 대역에 해당합니다. Cloud Volumes Service 의 경우 이는 클라우드 공급자 지역에 해당합니다. 각 스토리지 풀에는 성능 특성과 데이터 보호 특성을 정의하는 일련의 고유한 스토리지 속성이 있습니다.

여기의 다른 객체와 달리 스토리지 풀 후보는 항상 자동으로 검색되고 관리됩니다.

Trident `Volume` 사물

볼륨은 NFS 공유, iSCSI 및 FC LUN과 같은 백엔드 엔드포인트를 포함하는 기본 프로비저닝 단위입니다. Kubernetes에서는 이것이 직접적으로 대응합니다. `PersistentVolumes` . 볼륨을 생성할 때는 볼륨을 프로비저닝할 수 있는 위치와 크기를 결정하는 스토리지 클래스가 있는지 확인하세요.

> - Kubernetes에서는 이러한 객체가 자동으로 관리됩니다. Trident 제공한 내용을 확인할 수 있습니다.
> - 연관된 스냅샷이 있는 PV를 삭제하면 해당 Trident 볼륨이 삭제 상태로 업데이트됩니다. Trident 볼륨을 삭제하려면 볼륨의 스냅샷을 제거해야 합니다.

볼륨 구성은 프로비저닝된 볼륨이 가져야 하는 속성을 정의합니다.

기인하다	유형	필수의	설명
버전	끈	아니요	Trident API 버전("1")
이름	끈	예	생성할 볼륨의 이름
스토리지클래스	끈	예	볼륨을 프로비저닝할 때 사용할 스토리지 클래스
크기	끈	예	프로비저닝할 볼륨의 크기(바이트)
규약	끈	아니요	사용할 프로토콜 유형: "파일" 또는 "블록"
내부 이름	끈	아니요	저장 시스템의 객체 이름입니다. Trident 에서 생성됨
cloneSourceVolume	끈	아니요	ontap(nas, san) & solidfire-*: 복제할 볼륨의 이름
splitOnClone	끈	아니요	ontap(nas, san): 복제본을 부모로부터 분리합니다.
스냅샷 정책	끈	아니요	ontap-*: 사용할 스냅샷 정책
스냅샷예약	끈	아니요	ontap-*: 스냅샷을 위해 예약된 볼륨의 백분율
수출정책	끈	아니요	ontap-nas*: 사용할 정책 내보내기
스냅샷 디렉토리	부울	아니요	ontap-nas*: 스냅샷 디렉토리가 표시되는지 여부
unixPermissions	끈	아니요	ontap-nas*: 초기 UNIX 권한
블록 크기	끈	아니요	solidfire-*: 블록/섹터 크기
파일 시스템	끈	아니요	파일 시스템 유형

Trident 생성 `internalName` 볼륨을 생성할 때. 이는 두 단계로 구성됩니다. 첫째, 저장 접두사(기본값)를 추가합니다. `trident` 또는 백엔드 구성의 접두사)를 볼륨 이름에 추가하여 다음과 같은 형식의 이름을 생성합니다. `<prefix>-<volume-name>` . 그런 다음 백엔드에서 허용되지 않는 문자를 대체하여 이름을 정리합니다. ONTAP 백엔드의 경우 하이픈을 밑줄로 바꿉니다(따라서 내부 이름은 다음과 같습니다. `<prefix>_<volume-name>`). Element 백엔드의 경우 밑줄을 하이픈으로 바꿉니다.

REST API를 사용하여 볼륨 구성을 사용하여 볼륨을 직접 프로비저닝할 수 있지만 Kubernetes 배포에서는 대부분의 사용자가 표준 Kubernetes를 사용할 것으로 예상합니다. `PersistentVolumeClaim` 방법. Trident 프로비저닝 프로세스의 일부로 이 볼륨 객체를 자동으로 생성합니다.

Trident `Snapshot` 사물

스냅샷은 볼륨의 특정 시점 복사본으로, 새로운 볼륨을 프로비저닝하거나 상태를 복원하는 데 사용할 수 있습니다. Kubernetes에서는 이것이 직접적으로 대응합니다. `VolumeSnapshotContent` 사물. 각 스냅샷은 볼륨과 연결되며, 볼륨은 스냅샷의 데이터 소스입니다.

각 `Snapshot` 개체에는 아래 나열된 속성이 포함됩니다.

기인하다	유형	필수의	설명
버전	끈	예	Trident API 버전("1")
이름	끈	예	Trident 스냅샷 객체의 이름
내부 이름	끈	예	스토리지 시스템의 Trident 스냅샷 개체 이름
볼륨 이름	끈	예	스냅샷이 생성되는 영구 볼륨의 이름
볼륨 내부 이름	끈	예	스토리지 시스템의 연관된 Trident 볼륨 개체의 이름

Kubernetes에서는 이러한 객체가 자동으로 관리됩니다. Trident 제공한 내용을 확인할 수 있습니다.

쿠버네티스가 `VolumeSnapshot` 객체 요청이 생성되면 Trident 백업 스토리지 시스템에 스냅샷 객체를 생성하여 작동합니다. 그만큼 `internalName` 이 스냅샷 객체의 접두사를 결합하여 생성됩니다. `snapshot-` 와 함께 `UID` 의 `VolumeSnapshot` 객체(예를 들어, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` 그리고 `volumeInternalName` 백업 볼륨의 세부 정보를 얻어서 채워집니다.

Trident `ResourceQuota` 물체

Trident 데몬셋은 다음을 소모합니다. `system-node-critical` 우선순위 클래스는 Kubernetes에서 사용할 수 있는 가장 높은 우선순위 클래스로, Trident 가 노드를 정상적으로 종료하는 동안 볼륨을 식별하고 정리할 수 있도록 하며, Trident 데몬셋 포드가 리소스 압박이 높은 클러스터에서 우선순위가 낮은 워크로드를 우선적으로 처리할 수 있도록 합니다.

이를 달성하기 위해 Trident 다음을 사용합니다. `ResourceQuota` Trident 데몬셋에서 "시스템 노드에 중요한" 우선순위 클래스가 충족되는지 확인하기 위한 객체입니다. 배포 및 데몬셋 생성 전에 Trident 다음을 찾습니다. `ResourceQuota` 객체를 찾고, 발견되지 않으면 이를 적용합니다.

기본 리소스 할당량 및 우선 순위 클래스에 대한 더 많은 제어가 필요한 경우 다음을 생성할 수 있습니다. `custom.yaml` 또는 구성 `ResourceQuota` Helm 차트를 사용하여 객체를 생성합니다.

다음은 Trident 데몬셋의 우선순위를 지정하는 `ResourceQuota` 객체의 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 다음을 참조하세요."쿠버네티스: 리소스 할당량" .

정리하다 `ResourceQuota` 설치가 실패하면

설치가 실패하는 드문 경우 `ResourceQuota` 객체가 생성되었습니다. 첫 번째 시도"설치 제거" 그리고 다시 설치하세요.

그래도 작동하지 않으면 수동으로 제거하세요. `ResourceQuota` 물체.

제거하다 `ResourceQuota`

자신의 리소스 할당을 제어하려는 경우 Trident 를 제거할 수 있습니다. `ResourceQuota` 다음 명령을 사용하여 객체를 생성합니다.

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

Pod 보안 표준(PSS) 및 보안 컨텍스트 제약 조건(SCC)

Kubernetes Pod 보안 표준(PSS)과 Pod 보안 정책(PSP)은 권한 수준을 정의하고 Pod의 동작을 제한합니다. OpenShift 보안 컨텍스트 제약 조건(SCC)은 마찬가지로 OpenShift Kubernetes Engine에 특정한 포드 제한을 정의합니다. 이러한 사용자 정의 기능을 제공하기 위해 Trident 설치 중에 특정 권한을 부여합니다. 다음 섹션에서는 Trident 가 설정한 권한에 대해 자세히 설명합니다.

PSS는 Pod 보안 정책(PSP)을 대체합니다. PSP는 Kubernetes v1.21에서 더 이상 지원되지 않으며 v1.25에서 제거될 예정입니다. 자세한 내용은 다음을 참조하세요."쿠버네티스: 보안" .

필수 Kubernetes 보안 컨텍스트 및 관련 필드

허가	설명
특권	CSI에서는 마운트 지점이 양방향이어야 합니다. 즉, Trident 노드 포드는 권한이 있는 컨테이너를 실행해야 합니다. 자세한 내용은 다음을 참조하세요."Kubernetes: 마운트 전파" .
호스트 네트워킹	iSCSI 데몬에 필요합니다. `iscsiadm` iSCSI 마운트를 관리하고 호스트 네트워킹을 사용하여 iSCSI 데몬과 통신합니다.
호스트 IPC	NFS는 NFSD와 통신하기 위해 프로세스 간 통신(IPC)을 사용합니다.
호스트 PID	시작하려면 필요합니다 `rpc-statd` NFS용. Trident 호스트 프로세스를 쿼리하여 다음을 확인합니다. `rpc-statd` NFS 볼륨을 마운트하기 전에 실행 중입니다.
역량	그만큼 `SYS_ADMIN` 기능은 권한이 있는 컨테이너의 기본 기능의 일부로 제공됩니다. 예를 들어, Docker는 권한이 있는 컨테이너에 대해 다음과 같은 기능을 설정합니다. `CapPrm: 0000003fffffffff` `CapEff: 0000003fffffffff`
세콤프	Seccomp 프로필은 특권 컨테이너에서는 항상 "제한 없음" 상태이므로 Trident 에서 활성화할 수 없습니다.
셀리눅스	OpenShift에서는 권한이 있는 컨테이너가 실행됩니다. `spc_t` ("Super Privileged Container") 도메인에서 실행되고 권한이 없는 컨테이너는 `container_t` 도메인. ~에 `containerd` , 와 함께 `container-selinux` 설치되면 모든 컨테이너가 실행됩니다. `spc_t` SELinux를 효과적으로 비활성화하는 도메인입니다. 따라서 Trident 추가하지 않습니다. `seLinuxOptions` 컨테이너로.
DAC	권한이 있는 컨테이너는 루트로 실행해야 합니다. 권한이 없는 컨테이너는 CSI에 필요한 유닉스 소켓에 액세스하기 위해 루트로 실행됩니다.

포드 보안 표준(PSS)

상표	설명	기본
`pod-security.kubernetes.io/enforce` `pod-security.kubernetes.io/enforce-version`	Trident Controller와 노드가 설치 네임스페이스에 들어갈 수 있도록 허용합니다. 네임스페이스 레이블을 변경하지 마세요.	`enforce: privileged` `enforce-version: <version of the current cluster or highest version of PSS tested.>`

네임스페이스 레이블을 변경하면 포드가 예약되지 않거나, "생성 오류: …" 또는 "경고: trident-csi-…"가 발생할 수 있습니다. 이런 일이 발생하면 네임스페이스 레이블을 확인하십시오. `privileged` 변경되었습니다. 그렇다면 Trident 다시 설치하세요.

Pod 보안 정책(PSP)

필드	설명	기본
`allowPrivilegeEscalation`	권한이 있는 컨테이너는 권한 상승을 허용해야 합니다.	`true`
`allowedCSIDrivers`	Trident 인라인 CSI 임시 볼륨을 사용하지 않습니다.	비어 있는
`allowedCapabilities`	권한이 없는 Trident 컨테이너는 기본 설정보다 더 많은 기능을 필요로 하지 않으며 권한이 있는 컨테이너는 가능한 모든 기능을 부여받습니다.	비어 있는
`allowedFlexVolumes`	Trident 는 사용하지 않습니다"FlexVolume 드라이버" 따라서 허용되는 볼륨 목록에 포함되지 않습니다.	비어 있는
`allowedHostPaths`	Trident 노드 포드는 노드의 루트 파일 시스템을 마운트하므로 이 목록을 설정하는 데에는 이점이 없습니다.	비어 있는
`allowedProcMountTypes`	Trident 어떤 것도 사용하지 않습니다 `ProcMountTypes` .	비어 있는
`allowedUnsafeSysctls`	Trident 안전하지 않은 것을 요구하지 않습니다. `sysctls` .	비어 있는
`defaultAddCapabilities`	특권 컨테이너에는 기능을 추가할 필요가 없습니다.	비어 있는
`defaultAllowPrivilegeEscalation`	권한 확대 허용은 각 Trident Pod에서 처리됩니다.	`false`
`forbiddenSysctls`	아니요 `sysctls` 허용됩니다.	비어 있는
`fsGroup`	Trident 컨테이너는 루트로 실행됩니다.	`RunAsAny`
`hostIPC`	NFS 볼륨을 마운트하려면 호스트 IPC가 통신해야 합니다. `nfsd`	`true`
`hostNetwork`	iscsiadm은 호스트 네트워크가 iSCSI 데몬과 통신하는 데 필요합니다.	`true`
`hostPID`	호스트 PID는 다음을 확인하는 데 필요합니다. `rpc-statd` 노드에서 실행 중입니다.	`true`
`hostPorts`	Trident 호스트 포트를 사용하지 않습니다.	비어 있는
`privileged`	Trident 노드 포드는 볼륨을 마운트하기 위해 권한이 있는 컨테이너를 실행해야 합니다.	`true`
`readOnlyRootFilesystem`	Trident 노드 포드는 노드 파일 시스템에 써야 합니다.	`false`

필드	설명	기본
requiredDropCapabilities	Trident 노드 포드는 권한이 있는 컨테이너를 실행하며 기능을 삭제할 수 없습니다.	none
runAsGroup	Trident 컨테이너는 루트로 실행됩니다.	RunAsAny
runAsUser	Trident 컨테이너는 루트로 실행됩니다.	runAsAny
runtimeClass	Trident 사용하지 않습니다 RuntimeClasses .	비어 있는
seLinux	Trident 설정되지 않습니다 seLinuxOptions 현재 컨테이너 런타임과 Kubernetes 배포판이 SELinux를 처리하는 방식에 차이가 있기 때문입니다.	비어 있는
supplementalGroups	Trident 컨테이너는 루트로 실행됩니다.	RunAsAny
volumes	Trident 포드에는 이러한 볼륨 플러그인이 필요합니다.	hostPath, projected, emptyDir

보안 컨텍스트 제약 조건(SCC)

라벨	설명	기본
allowHostDirVolumePlugin	Trident 노드 포드는 노드의 루트 파일 시스템을 마운트합니다.	true
allowHostIPC	NFS 볼륨을 마운트하려면 호스트 IPC가 통신해야 합니다. nfsd .	true
allowHostNetwork	iscsiadm은 호스트 네트워크가 iSCSI 데몬과 통신하는 데 필요합니다.	true
allowHostPID	호스트 PID는 다음을 확인하는 데 필요합니다. rpc-statd 노드에서 실행 중입니다.	true
allowHostPorts	Trident 호스트 포트를 사용하지 않습니다.	false
allowPrivilegeEscalation	권한이 있는 컨테이너는 권한 상승을 허용해야 합니다.	true
allowPrivilegedContainer	Trident 노드 포드는 볼륨을 마운트하기 위해 권한이 있는 컨테이너를 실행해야 합니다.	true
allowedUnsafeSysctls	Trident 안전하지 않은 것을 요구하지 않습니다. sysctls .	none

라벨	설명	기본
`allowedCapabilities`	권한이 없는 Trident 컨테이너는 기본 설정보다 더 많은 기능을 필요로 하지 않으며 권한이 있는 컨테이너는 가능한 모든 기능을 부여받습니다.	비어 있는
`defaultAddCapabilities`	특권 컨테이너에는 기능을 추가할 필요가 없습니다.	비어 있는
`fsGroup`	Trident 컨테이너는 루트로 실행됩니다.	`RunAsAny`
`groups`	이 SCC는 Trident 에만 적용되며 사용자에게 적용됩니다.	비어 있는
`readOnlyRootFilesystem`	Trident 노드 포드는 노드 파일 시스템에 써야 합니다.	`false`
`requiredDropCapabilities`	Trident 노드 포드는 권한이 있는 컨테이너를 실행하며 기능을 삭제할 수 없습니다.	`none`
`runAsUser`	Trident 컨테이너는 루트로 실행됩니다.	`RunAsAny`
`seLinuxContext`	Trident 설정되지 않습니다 `seLinuxOptions` 현재 컨테이너 런타임과 Kubernetes 배포판이 SELinux를 처리하는 방식에 차이가 있기 때문입니다.	비어 있는
`seccompProfiles`	특권 컨테이너는 항상 "제한되지 않음"으로 실행됩니다.	비어 있는
`supplementalGroups`	Trident 컨테이너는 루트로 실행됩니다.	`RunAsAny`
`users`	Trident 네임스페이스에서 이 SCC를 Trident 사용자에 바인딩하기 위해 하나의 항목이 제공됩니다.	해당 없음
`volumes`	Trident 포드에는 이러한 볼륨 플러그인이 필요합니다.	`hostPath, downwardAPI, projected, emptyDir`