참조하십시오

Astra Trident

NetApp
November 14, 2025

목차

참조하십시오

Astra Trident 포트

Astra Trident가 통신에 사용하는 포트에 대해 자세히 알아보십시오.

Astra Trident 포트

Astra Trident는 다음 포트를 통해 통신합니다.

포트	목적
8443	백채널 HTTPS
8001입니다	Prometheus 메트릭 엔드포인트
8000	Trident REST 서버
17546	Trident 디포드에 사용되는 활성/준비 프로브 포트

을 사용하여 설치하는 동안 활성/준비 프로브 포트를 변경할 수 있습니다 `--probe-port` 깃발. 이 포트가 작업자 노드의 다른 프로세스에서 사용되지 않도록 하는 것이 중요합니다.

Astra Trident REST API

있습니다 "tridentctl 명령 및 옵션" Astra Trident REST API와 상호 작용하는 가장 쉬운 방법은 REST 엔드포인트를 직접 사용하는 것입니다.

REST API 사용 시기

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

더 나은 보안을 위해 Astra Trident를 사용해 보십시오 `REST API` 는 POD 내에서 실행할 때 기본적으로 localhost로 제한됩니다. 이 동작을 변경하려면 Astra Trident를 설정해야 합니다 `-address` 인수가 해당 POD 구성의 인수입니다.

REST API 사용

API는 다음과 같이 작동합니다.

`GET`

- `GET <trident-address>/trident/v1/<object-type>`: 해당 형식의 모든 개체를 나열합니다.
- `GET <trident-address>/trident/v1/<object-type>/<object-name>`(Control 에서 상속됨): 명명된 개체의 세부 정보를 가져옵니다.

`POST`

`POST <trident-address>/trident/v1/<object-type>`: 지정된 형식의 개체를 만듭니다.

- 생성할 개체의 JSON 구성이 필요합니다. 각 개체 유형의 사양은 링크 tridentctl.html 을 참조하십시오 [`tridentctl` 명령 및 옵션].
- 개체가 이미 있는 경우 동작이 달라집니다. backends는 기존 개체를 업데이트하지만 다른 모든 개체 형식은 작업에 실패합니다.

`DELETE`

`DELETE <trident-address>/trident/v1/<object-type>/<object-name>`: 명명된 리소스를 삭제합니다.

> 백엔드 또는 스토리지 클래스와 연결된 볼륨은 계속 존재하므로 별도로 삭제해야 합니다. 자세한 내용은 링크: tridentctl.html 을 참조하십시오[`tridentctl` 명령 및 옵션].

이러한 API의 호출 방법에 대한 예제를 보려면 디버그를 전달합니다 (`-d`) 깃발. 자세한 내용은 링크: tridentctl.html 을 참조하십시오[`tridentctl` 명령 및 옵션].

명령줄 옵션

Astra Trident는 Trident Orchestrator를 위한 여러 명령줄 옵션을 제공합니다. 이러한 옵션을 사용하여 배포를 수정할 수 있습니다.

로깅

- `-debug`: 디버깅 출력을 활성화합니다.
- `-loglevel <level>`: 로깅 수준을 설정합니다(debug, info, warn, error, fatal). 기본적으로 info(정보) 가 사용됩니다.

쿠버네티스

- `-k8s_pod`: 이 옵션 또는 을 사용합니다 `-k8s_api_server` Kubernetes 지원을 사용하도록 설정하십시오. 이 설정을 사용하면 Trident에서 포함된 POD의 Kubernetes 서비스 계정 자격 증명을 사용하여 API 서버에 연락합니다. Trident가 서비스 계정이 활성화된 Kubernetes 클러스터에서 POD로 실행되는 경우에만 작동합니다.
- `-k8s_api_server <insecure-address:insecure-port>`: 이 옵션 또는 을 사용합니다 `-k8s_pod` Kubernetes 지원을 사용하도록 설정하십시오. Trident가 지정된 경우 제공된 비보안 주소 및 포트를 사용하여 Kubernetes API 서버에 연결합니다. 따라서 Trident를 POD 외부에 배포할 수 있지만 API 서버에 대한 비보안 연결만 지원합니다. 를 사용하여 포드에 Trident를 배포하여 안전하게 연결합니다 `-k8s_pod` 옵션을 선택합니다.
- `-k8s_config_path <file>`: 필수. KubeConfig 파일에 대한 이 경로를 지정해야 합니다.

Docker 를 참조하십시오

- `-volume_driver <name>`: Docker 플러그인을 등록할 때 사용되는 드라이버 이름입니다. 기본값은 입니다 `netapp`.
- `-driver_port <port-number>`: UNIX 도메인 소켓이 아닌 이 포트에서 수신 대기하십시오.

- `-config <file>`필수: 백엔드 구성 파일에 대한 이 경로를 지정해야 합니다.

휴식

- `-address <ip-or-host>`: Trident의 REST 서버가 수신할 주소를 지정합니다. 기본값은 localhost입니다. localhost에서 듣거나 Kubernetes Pod에서 실행 중인 경우, REST 인터페이스는 Pod 외부에서 직접 액세스할 수 없습니다. 사용 `-address ""` REST 인터페이스를 POD IP 주소에서 액세스할 수 있도록 합니다.

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

- `-port <port-number>`: Trident의 REST 서버가 수신할 포트를 지정합니다. 기본값은 8000입니다.
- `-rest`: REST 인터페이스를 활성화합니다. 기본값은 true 입니다.

Kubernetes와 통합된 NetApp 제품

NetApp 스토리지 제품 포트폴리오는 Kubernetes 클러스터의 다양한 측면과 통합되어 고급 데이터 관리 기능을 제공하며, Kubernetes 구축의 기능, 기능, 성능 및 가용성을 향상합니다.

아스트라

"아스트라" 퍼블릭 클라우드와 온프레미스 모두에서 Kubernetes에서 실행되는 데이터 리치 컨테이너 워크로드를 손쉽게 관리, 보호 및 이동할 수 있습니다. Astra는 퍼블릭 클라우드 및 온프레미스에 있는 NetApp의 검증되고 광범위한 스토리지 포트폴리오의 Trident를 사용하여 영구 컨테이너 스토리지를 프로비저닝하고 제공합니다. 또한, 스냅샷, 백업 및 복원, 활동 로그, 데이터 보호, 재해/데이터 복구, 데이터 감사, Kubernetes 워크로드의 마이그레이션 사용 사례를 위한 액티브 클론 복제와 같은 풍부한 고급 애플리케이션 인식 데이터 관리 기능을 제공합니다.

ONTAP

ONTAP는 모든 애플리케이션에 고급 데이터 관리 기능을 제공하는 NetApp의 다중 프로토콜 통합 스토리지 운영 체제입니다. ONTAP 시스템은 All-Flash, 하이브리드 또는 All-HDD 구성을 제공하며 엔지니어링 하드웨어(FAS 및 AFF), 화이트박스(ONTAP Select), 클라우드 전용(Cloud Volumes ONTAP) 등 다양한 구축 모델을 제공합니다.

> Trident는 위에 언급된 모든 ONTAP 배포 모델을 지원합니다.

Cloud Volumes ONTAP

"Cloud Volumes ONTAP" 는 클라우드에서 ONTAP 데이터 관리 소프트웨어를 실행하는 소프트웨어 전용 스토리지 어플라이언스입니다. Cloud Volumes ONTAP를 운영 워크로드, 재해 복구, DevOps, 파일 공유 및 데이터베이스 관리에 사용할 수 있습니다. 스토리지 효율성, 고가용성, 데이터 복제, 데이터 계층화, 애플리케이션 정합성을 보장함으로써 엔터프라이즈 스토리지를 클라우드로 확장합니다.

NetApp ONTAP용 Amazon FSx

"NetApp ONTAP용 Amazon FSx" 은 완벽하게 관리되는 AWS 서비스로, 고객이 NetApp의 ONTAP 스토리지 운영 체제에서 제공하는 파일 시스템을 시작하고 실행할 수 있도록 지원합니다. ONTAP용 FSX를 사용하면 고객이 익숙한 NetApp 기능, 성능 및 관리 기능을 활용하는 동시에 AWS에 데이터를 저장할 때 간편성, 민첩성, 보안, 확장성을 활용할 수 있습니다. ONTAP용 FSX는 ONTAP의 다양한 파일 시스템 기능과 관리 API를 지원합니다.

Element 소프트웨어

"요소" 스토리지 관리자가 성능을 보장하고 단순화된 스토리지 설치 공간을 활용하여 워크로드를 통합할 수 있도록 지원합니다. Element는 스토리지 관리의 모든 측면을 자동화하는 API와 결합되어 스토리지 관리자가 더 적은 노력으로 더 많은 작업을 수행할 수 있게 합니다.

NetApp HCI

"NetApp HCI" 일상적인 작업을 자동화하고 인프라 관리자가 보다 중요한 기능에 집중할 수 있도록 하여 데이터 센터의 관리 및 확장을 단순화합니다.

NetApp HCI는 Trident에서 완전히 지원합니다. Trident는 컨테이너화된 애플리케이션에 대한 스토리지 장치를 기본 NetApp HCI 스토리지 플랫폼에 직접 프로비저닝 및 관리할 수 있습니다.

Azure NetApp Files

"Azure NetApp Files" NetApp에서 제공하는 엔터프라이즈급 Azure 파일 공유 서비스입니다. Azure에서 기본적으로 가장 까다로운 파일 기반 워크로드를 실행하고 NetApp에서 기대하는 성능 및 강력한 데이터 관리를 제공할 수 있습니다.

Google Cloud용 Cloud Volumes Service

"Google Cloud용 NetApp Cloud Volumes Service" 는 NFS 및 SMB를 통해 NAS 볼륨을 All-Flash 성능으로 제공하는 클라우드 네이티브 파일 서비스입니다. 이 서비스를 사용하면 기존 애플리케이션을 포함한 모든 워크로드를 GCP 클라우드에서 실행할 수 있습니다. GCE(Google Compute Engine) 인스턴스에 대한 일관된 고성능, 즉각적인 복제, 데이터 보호 및 보안 액세스를 제공하는 완전 관리형 서비스를 제공합니다.

Kubernetes 및 Trident 오브젝트

REST API를 사용하여 리소스 객체를 읽고 쓰면서 Kubernetes 및 Trident와 상호 작용할 수 있습니다. Kubernetes 및 Trident, Trident 및 스토리지와 Kubernetes 및 스토리지 간의 관계를 결정하는 몇 가지 리소스 개체가 있습니다. 이러한 오브젝트 중 일부는 Kubernetes를 통해 관리되며 나머지는 Trident를 통해 관리됩니다.

개체가 서로 어떻게 상호 작용합니까?

오브젝트, 목표 및 상호 작용 방식을 이해하는 가장 쉬운 방법은 Kubernetes 사용자의 스토리지에 대한 단일 요청을 따르는 것입니다.

1. 사용자가 를 만듭니다 `PersistentVolumeClaim` 신규 요청 중 `PersistentVolume` Kubernetes를 사용할 경우 특정 크기입니다 `StorageClass` 이전에 관리자가 구성한 것입니다.
2. Kubernetes를 `StorageClass` Trident를 프로비저닝자로 식별하고 Trident에 요청된 클래스에 대한 볼륨을 프로비저닝하는 방법을 알려주는 매개 변수를 포함합니다.
3. Trident가 독자적인 모양을 하고 있습니다 `StorageClass` 일치하는 항목을 식별하는 동일한 이름을 사용합니다 `Backends` 및 `StoragePools` 볼륨을 프로비저닝하는 데 사용할 수 있다는 것을 알 수 있습니다.
4. Trident가 일치하는 백엔드에 스토리지를 프로비저닝하고 두 개의 오브젝트, 즉 A를 생성합니다 `PersistentVolume` Kubernetes에서 볼륨 찾기, 마운트 및 처리 방법과 Trident의 볼륨을 사용하여 간의 관계를 유지하는 방법을 Kubernetes에서 알려줍니다 `PersistentVolume` 및 실제 스토리지를 누릅니다.

5. Kubernetes에서 에 바인딩합니다 `PersistentVolumeClaim` 새로운 제품으로 `PersistentVolume`. 를 포함하는 Pod입니다 `PersistentVolumeClaim` 실행 중인 모든 호스트에 PersistentVolume을 마운트합니다.
6. 사용자가 를 만듭니다 `VolumeSnapshot` 를 사용하여 기존 PVC의 `VolumeSnapshotClass` Trident를 가리키죠.
7. Trident는 PVC와 연결된 볼륨을 식별하고 백엔드에 볼륨의 스냅샷을 생성합니다. 또한 을 생성합니다 `VolumeSnapshotContent` 이 방법은 Kubernetes에서 스냅샷을 식별하는 방법을 설명합니다.
8. 사용자는 을 만들 수 있습니다 `PersistentVolumeClaim` 사용 `VolumeSnapshot` 소스로 사용합니다.
9. Trident는 필요한 스냅샷을 식별하고 과 관련된 동일한 단계를 수행합니다 `PersistentVolume` 그리고 A `Volume`.

Kubernetes 객체에 대한 자세한 내용은 를 읽는 것이 좋습니다 "영구 볼륨" 섹션을 참조하십시오.

쿠버네티스 `PersistentVolumeClaim` 오브젝트

쿠버네티스 `PersistentVolumeClaim` 개체는 Kubernetes 클러스터 사용자가 만든 스토리지 요청입니다.

Trident는 표준 사양 외에도 사용자가 백엔드 구성에서 설정한 기본값을 무효화하려는 경우 다음 볼륨별 주석을 지정할 수 있도록 합니다.

주석	볼륨 옵션	지원되는 드라이버
trident.netapp.io/fileSystem	파일 시스템	ONTAP-SAN, solidfire-SAN, ONTAP-SAN - 경제성
trident.netapp.io/cloneFromPVC	CloneSourceVolume	ONTAP-NAS, ONTAP-SAN, solidfire-SAN, Azure-NetApp-파일, GCP-CV, ONTAP-SAN - 경제성
trident.netapp.io/splitOnClone	SplitOnClone 을 참조하십시오	ONTAP-NAS, ONTAP-SAN
trident.netapp.io/protocol	프로토콜	모두
trident.netapp.io/exportPolicy	내보내기 정책	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup
trident.netapp.io/snapshotPolicy	스냅샷 정책	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup, ONTAP-SAN
trident.netapp.io/snapshotReserve	snapshotReserve	ONTAP-NAS, ONTAP-NAS-flexgroup, ONTAP-SAN, GCP-CV
trident.netapp.io/snapshotDirectory	스냅샷 디렉토리	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup
trident.netapp.io/blockSize	블록 크기	solidfire-SAN

생성된 PV에 가 있는 경우 `Delete` 부가세 반환 청구액 정책인 Trident는 PV가 릴리스될 때(즉, 사용자가 PVC를 삭제할 때) PV와 보조 볼륨을 모두 삭제합니다. 삭제 작업이 실패할 경우 Trident는 PV를 해당 상태로 표시하고 성공할 때까지 또는 PV를 수동으로 삭제할 때까지 주기적으로 작업을 다시 시도합니다. PV에서 를 사용하는 경우 `Retain` Trident는 이 정책을 무시하고 관리자가 Kubernetes 및 백엔드에서 이를 정리하여 제거하기 전에 볼륨을 백업 또는

검사할 수 있다고 가정합니다. PV를 삭제해도 Trident에서 백업 볼륨을 삭제하지 않습니다. REST API를 사용하여 제거해야 합니다 (`tridentctl`)를 클릭합니다.

Trident는 CSI 사양을 사용하여 볼륨 스냅샷 생성을 지원합니다. 볼륨 스냅샷을 생성하고 이를 데이터 소스로 사용하여 기존 PVC를 복제할 수 있습니다. 이렇게 하면 PVS의 시점 복제본을 스냅샷 형태로 Kubernetes에 표시할 수 있습니다. 그런 다음 스냅샷을 사용하여 새 PVS를 생성할 수 있습니다. 한 번 살펴보십시오 `On-Demand Volume Snapshots` 어떻게 작동되는지 확인하십시오.

Trident는 도 제공합니다 `cloneFromPVC` 및 `splitOnClone` 클론 생성을 위한 주석. 이러한 주석을 사용하여 Kubernetes 1.13 이하 버전에서 CSI 구현을 사용하지 않고 PVC를 복제하거나 Kubernetes 릴리스가 베타 볼륨 스냅샷(Kubernetes 1.16 이하)을 지원하지 않을 경우 사용할 수 있습니다. Trident 19.10은 PVC를 통한 클론 생성을 위한 CSI 워크플로우를 지원합니다.

> ⓘ 를 사용할 수 있습니다 `cloneFromPVC` 및 `splitOnClone` CSI Trident 및 기존 비 CSI 프런트엔드 주석이 포함된 주석.

다음은 사용자가 이미 PVC를 호출한 경우입니다 `mysql`에서 라는 새 `PVC`를 생성할 수 있습니다 `mysqlclone` 과 같은 주석을 사용합니다 `trident.netapp.io/cloneFromPVC: mysql`. 이 주석을 설정하면 Trident가 볼륨을 처음부터 프로비저닝하는 대신 MySQL PVC에 해당하는 볼륨을 클론합니다.

다음 사항을 고려하십시오.

- 유휴 볼륨의 클론을 생성하는 것이 좋습니다.
- PVC와 그 클론은 동일한 Kubernetes 네임스페이스에서 동일한 스토리지 클래스를 가져야 합니다.
- 를 사용하여 `ontap-nas` 및 `ontap-san` 드라이버는 PVC 주석을 설정하는 것이 좋습니다 `trident.netapp.io/splitOnClone` 와 함께 제공됩니다 `trident.netapp.io/cloneFromPVC`. 와 함께 `trident.netapp.io/splitOnClone` 를 로 설정합니다 `true`Trident는 상위 볼륨에서 복제된 볼륨을 분할하여 복제된 볼륨의 라이프사이클을 완전히 상위 볼륨에서 분리함으로써 일부 스토리지 효율성을 잃지 않습니다. 설정 안 합니다 `trident.netapp.io/splitOnClone` 또는 로 설정합니다 `false` 상위 볼륨과 클론 볼륨 간의 종속성을 생성하여 백엔드에서 공간 소비를 줄임으로써 클론이 먼저 삭제되지 않는 한 상위 볼륨을 삭제할 수 없습니다. 클론을 분할하는 것이 올바른 시나리오는 빈 데이터베이스 볼륨을 복제하여 볼륨과 해당 클론이 크게 달라질 것으로 예상되며 ONTAP에서 제공하는 스토리지 효율성의 이점을 얻지 못하는 경우입니다.

를 클릭합니다 `sample-input` 디렉토리에는 Trident와 함께 사용할 PVC 정의의 예가 포함되어 있습니다. Trident 볼륨과 관련된 매개 변수 및 설정에 대한 자세한 설명은 Trident Volume 개체를 참조하십시오.

쿠버네티스 `PersistentVolume` 오브젝트

쿠버네티스 `PersistentVolume` 객체는 Kubernetes 클러스터에서 사용할 수 있는 스토리지 부분을 나타냅니다. 사용 포드와 독립적인 라이프 사이클이 있습니다.

> ⓘ Trident가 작성합니다 `PersistentVolume` 제공하는 볼륨을 기반으로 하여 Kubernetes 클러스터에 자동으로 개체를 등록하고 등록합니다. 스스로 관리할 수 없습니다.

Trident를 참조하는 PVC를 만들 때 `StorageClass`, Trident는 해당 저장소 클래스를 사용하여 새 볼륨을 프로비저닝하고 해당 볼륨에 대한 새 PV를 등록합니다. 프로비저닝 볼륨과 해당 PV를 구성할 때 Trident는 다음 규칙을 따릅니다.

- Trident는 Kubernetes의 PV 이름과 스토리지 프로비저닝에 사용되는 내부 이름을 생성합니다. 두 경우 모두 이름은 해당 범위에서 고유합니다.

- 볼륨의 크기는 플랫폼에 따라 가장 가까운 할당 가능한 수량으로 반올림될 수 있지만 PVC에서 요청된 크기와 최대한 가깝게 일치합니다.

쿠버네티스 `StorageClass` 오브젝트

쿠버네티스 `StorageClass` 오브젝트는 의 이름으로 지정됩니다 `PersistentVolumeClaims` 속성 집합을 사용하여 스토리지를 프로비저닝합니다. 스토리지 클래스 자체는 사용할 구축 소유자를 식별하고 프로비저닝이 이해할 수 있는 조건으로 해당 자산 세트를 정의합니다.

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

쿠버네티스 `StorageClass` Trident를 사용하는 개체의 모양은 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 매개 변수는 Trident에만 해당되며 Trident에 클래스에 볼륨을 프로비저닝하는 방법을 알려줍니다.

스토리지 클래스 매개 변수는 다음과 같습니다.

속성	유형	필수 요소입니다	설명
속성	[string] 문자열을 매핑합니다	아니요	아래의 특성 섹션을 참조하십시오
스토리지 풀	Map [string] StringList 입니다	아니요	내의 스토리지 풀 목록에 백엔드 이름 매핑
추가 StoragePools	Map [string] StringList 입니다	아니요	내의 스토리지 풀 목록에 백엔드 이름 매핑
excludeStoragePools를 참조하십시오	Map [string] StringList 입니다	아니요	내의 스토리지 풀 목록에 백엔드 이름 매핑

스토리지 속성 및 가능한 값은 스토리지 풀 선택 특성 및 Kubernetes 속성으로 분류할 수 있습니다.

스토리지 풀 선택 특성입니다

이러한 매개 변수는 지정된 유형의 볼륨을 프로비저닝하는 데 사용해야 하는 Trident 관리 스토리지 풀을 결정합니다.

속성	유형	값	제공합니다	요청하십시오	에 의해 지원됩니다
미디어 [1]	문자열	HDD, 하이브리드, SSD	풀에는 이 유형의 미디어가 포함되어 있으며, 하이브리드는 둘 모두를 의미합니다	지정된 미디어 유형입니다	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup, ONTAP-SAN, solidfire-SAN
프로비저닝 유형	문자열	얇고 두껍습니다	풀은 이 프로비저닝 방법을 지원합니다	프로비저닝 방법이 지정되었습니다	Thick: All ONTAP; Thin: All ONTAP & solidfire-SAN
백엔드 유형	문자열	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup, ONTAP-SAN, solidfire-SAN, GCP-CV, Azure-NetApp-파일, ONTAP-SAN-이코노미	풀이 이 백엔드 유형에 속합니다	백엔드가 지정되었습니다	모든 드라이버
스냅샷 수	불입니다	참, 거짓	풀은 스냅샷이 있는 볼륨을 지원합니다	스냅샷이 활성화된 볼륨	ONTAP-NAS, ONTAP-SAN, solidfire-SAN, GCP-CV
복제	불입니다	참, 거짓	풀은 볼륨 클론을 지원합니다	클론이 활성화된 볼륨	ONTAP-NAS, ONTAP-SAN, solidfire-SAN, GCP-CV
암호화	불입니다	참, 거짓	풀은 암호화된 볼륨을 지원합니다	암호화가 활성화된 볼륨입니다	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroups, ONTAP-SAN
IOPS	내부	양의 정수입니다	풀은 이 범위에서 IOPS를 보장할 수 있습니다	볼륨은 이러한 IOPS를 보장합니다	solidfire-SAN

[1]: ONTAP Select 시스템에서 지원되지 않습니다

대부분의 경우 요청된 값이 프로비저닝에 직접적인 영향을 미치며, 예를 들어 일반 프로비저닝을 요청하면 볼륨이 걸쭉하게 프로비저닝됩니다. 하지만 Element 스토리지 풀은 제공된 IOPS 최소 및 최대값을 사용하여 요청된 값이 아닌 QoS 값을 설정합니다. 이 경우 요청된 값은 스토리지 풀을 선택하는 데만 사용됩니다.

을 사용하는 것이 가장 좋습니다 `attributes` 단독으로 특정 클래스의 요구사항을 충족하는 데 필요한 스토리지의

품질을 모델링합니다. Trident는 의 _ALL_과 일치하는 스토리지 풀을 자동으로 검색하여 선택합니다 `attributes` 지정할 수 있습니다.

을(를) 사용할 수 없는 경우 `attributes` 클래스에 맞는 풀을 자동으로 선택하려면 를 사용할 수 있습니다 `storagePools` 및 `additionalStoragePools` 풀을 더 세분화하거나 특정 풀 세트를 선택하기 위한 매개 변수입니다.

를 사용할 수 있습니다 `storagePools` 매개 변수를 사용하여 지정된 모든 풀과 일치하는 풀 세트를 추가로 제한합니다 `attributes`. 즉, Trident는 로 식별된 풀의 교차를 사용합니다 `attributes` 및 `storagePools` 프로비저닝에 필요한 매개 변수입니다. 매개 변수만 사용하거나 둘 다 함께 사용할 수 있습니다.

를 사용할 수 있습니다 `additionalStoragePools` 에서 선택한 풀에 관계없이 Trident가 프로비저닝에 사용하는 풀 세트를 확장하는 매개 변수입니다 `attributes` 및 `storagePools` 매개 변수.

를 사용할 수 있습니다 `excludeStoragePools` Trident가 프로비저닝에 사용하는 풀 세트를 필터링하는 매개 변수입니다. 이 매개 변수를 사용하면 일치하는 풀이 모두 제거됩니다.

에 있습니다 `storagePools` 및 `additionalStoragePools` 매개 변수, 각 항목은 폼을 사용합니다 `<backend>:<storagePoolList>`, 위치 `<storagePoolList>` 는 지정된 백엔드에 대해 쉼표로 구분된 스토리지 풀 목록입니다. 예를 들어, 의 값을 입력합니다 `additionalStoragePools` 있을 것입니다 `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`. 이러한 목록에는 백엔드 및 목록 값 모두에 대한 regex 값이 적용됩니다. 을 사용할 수 있습니다 `tridentctl get backend` 백엔드 및 해당 풀의 목록을 가져옵니다.

Kubernetes 특성

이러한 특성은 동적 프로비저닝 중 Trident가 스토리지 풀/백엔드를 선택하는 데 아무런 영향을 주지 않습니다. 대신 이러한 특성은 Kubernetes 영구 볼륨에서 지원하는 매개 변수만 제공합니다. 작업자 노드는 파일 시스템 생성 작업을 담당하며 xfsprogs와 같은 파일 시스템 유틸리티가 필요할 수 있습니다.

속성	유형	값	설명	관련 드라이버	**Kubernetes** 버전
fsType입니다	문자열	ext4, ext3, xfs 등	블록 볼륨의 파일 시스템 유형입니다	solidfire-SAN, ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup, ONTAP-SAN, ONTAP-SAN - 경제성	모두
allowVolumeExpansion	부울	참, 거짓	PVC 크기 증가에 대한 지원을 활성화 또는 비활성화합니다	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup, ONTAP-SAN, ONTAP-SAN-이코노미, solidfire-SAN, GCP-CV, Azure-NetApp-파일	1.11+

속성	유형	값	설명	관련 드라이버	**Kubernetes** 버전
볼륨BindingMode를 선택합니다	문자열	Immediate, WaitForFirstConsumer입니다	볼륨 바인딩 및 동적 프로비저닝이 수행될 시기를 선택합니다	모두	1.19-1.26

- 를 클릭합니다 `fsType` 매개 변수는 SAN LUN에 대해 원하는 파일 시스템 유형을 제어하는 데 사용됩니다. 또한 Kubernetes는 의 존재 여부를 사용합니다 `fsType` 파일 시스템이 있음을 나타내는 스토리지 클래스에 있습니다. 볼륨 소유권은 를 사용하여 제어할 수 있습니다 `fsGroup` POD의 보안 컨텍스트는 에만 해당됩니다 `fsType` 가 설정됩니다. 을 참조하십시오 "Kubernetes: Pod 또는 컨테이너의 보안 컨텍스트를 구성합니다" 를 사용하여 볼륨 소유권을 설정하는 방법에 대한 개요를 보려면 를 참조하십시오 `fsGroup` 상황. Kubernetes가 에 적용됩니다 `fsGroup` 다음 경우에만 값:

  - `fsType` 스토리지 클래스에서 설정됩니다.
  - PVC 액세스 모드는 RWO입니다.

  NFS 스토리지 드라이버의 경우 파일 시스템이 NFS 내보내기의 일부로 이미 존재합니다. 를 사용합니다 `fsGroup` 스토리지 클래스는 여전히 을 지정해야 합니다 `fsType`. 로 설정할 수 있습니다 `nfs` 또는 null이 아닌 값을 입력합니다.

- 을 참조하십시오 "볼륨 확장" 볼륨 확장에 대한 자세한 내용은 를 참조하십시오.
- Trident 설치 프로그램 번들에는 의 Trident와 함께 사용할 수 있는 여러 가지 예제 스토리지 클래스 정의가 제공됩니다 `sample-input/storage-class-*.yaml`. Kubernetes 스토리지 클래스를 삭제하면 해당 Trident 스토리지 클래지도 삭제됩니다.

쿠버네티스 `VolumeSnapshotClass` 오브젝트

쿠버네티스 `VolumeSnapshotClass` 개체는 과 유사합니다 `StorageClasses`. 이 기능을 사용하면 여러 스토리지 클래스를 정의할 수 있으며, 스냅샷을 필요한 스냅샷 클래스와 연결하기 위해 볼륨 스냅숏에서 참조할 수 있습니다. 각 볼륨 스냅샷은 단일 볼륨 스냅샷 클래스와 연결됩니다.

A `VolumeSnapshotClass` 스냅샷을 생성하려면 관리자가 정의해야 합니다. 볼륨 스냅샷 클래스는 다음과 같은 정의로 생성됩니다.

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

를 클릭합니다 `driver` 의 볼륨 스냅샷을 요청하는 Kubernetes를 지정합니다 `csi-snapclass` 클래스는 Trident에서 처리합니다. 를 클릭합니다 `deletionPolicy` 스냅샷을 삭제해야 할 때 수행할 작업을 지정합니다. 시기 `deletionPolicy` 가 로 설정되어 있습니다 `Delete`스냅샷이 삭제되면 스토리지 클러스터의 기본 스냅샷 및 볼륨 스냅샷 객체가 제거됩니다. 또는 로 설정합니다 `Retain` 은 를 의미합니다 `VolumeSnapshotContent` 물리적 스냅샷이 보존됩니다.

쿠버네티스 VolumeSnapshot 오브젝트

쿠버네티스 VolumeSnapshot object는 볼륨의 스냅샷을 생성하는 요청입니다. PVC는 사용자가 볼륨에 대해 요청하는 것처럼 볼륨 스냅샷은 사용자가 기존 PVC의 스냅샷을 생성하도록 요청하는 것입니다.

볼륨 스냅샷 요청이 들어오면 Trident는 백엔드의 볼륨에 대한 스냅샷 생성을 자동으로 관리하고 고유한 를 생성하여 스냅샷을 표시합니다
VolumeSnapshotContent 오브젝트. 기존 PVC에서 스냅샷을 생성하고 새 PVC를 생성할 때 스냅샷을 DataSource로 사용할 수 있습니다.

VolumeSnapshot의 생수는 소스 PVC와는 독립적입니다. 소스 PVC가 삭제된 후에도 스냅샷이 지속됩니다. 연관된 스냅샷이 있는 PVC를 삭제할 때 Trident는 이 PVC에 대한 백업 볼륨을 * Deleting * 상태로 표시하지만 완전히 제거하지는 않습니다. 연결된 모든 스냅샷이 삭제되면 볼륨이 제거됩니다.

쿠버네티스 VolumeSnapshotContent 오브젝트

쿠버네티스 VolumeSnapshotContent 개체는 이미 프로비저닝된 볼륨에서 생성된 스냅샷을 나타냅니다. 이는 와 유사합니다 PersistentVolume 및 은 스토리지 클러스터에서 프로비저닝된 스냅샷을 나타냅니다. 과 유사합니다 PersistentVolumeClaim 및 PersistentVolume 객체, 스냅샷이 생성될 때 VolumeSnapshotContent 개체는 에 대한 일대일 매핑을 유지합니다 VolumeSnapshot 스냅샷 생성을 요청한 객체입니다.

Trident가 작성합니다 VolumeSnapshotContent 제공하는 볼륨을 기반으로 하여 Kubernetes 클러스터에 자동으로 개체를 등록하고 등록합니다. 스스로 관리할 수 없습니다.

를 클릭합니다 VolumeSnapshotContent 객체에는 과 같이 스냅샷을 고유하게 식별하는 세부 정보가 포함되어 있습니다 snapshotHandle. 여기 snapshotHandle 은 PV의 이름과 의 이름을 고유하게 조합한 것입니다 VolumeSnapshotContent 오브젝트.

스냅샷 요청이 들어오면 Trident가 백엔드에 스냅샷을 생성합니다. 스냅샷이 생성되면 Trident에서 을 구성합니다 VolumeSnapshotContent Kubernetes API에 스냅샷을 노출합니다.

쿠버네티스 CustomResourceDefinition 오브젝트

Kubernetes 사용자 지정 리소스는 관리자가 정의하며 비슷한 객체를 그룹화하는 데 사용되는 Kubernetes API의 엔드포인트입니다. Kubernetes에서는 오브젝트 컬렉션을 저장하기 위한 사용자 지정 리소스의 생성을 지원합니다. 를 실행하여 이러한 리소스 정의를 가져올 수 있습니다 kubectl get crds.

사용자 정의 리소스 정의(CRD) 및 관련 오브젝트 메타데이터는 Kubernetes에서 메타데이터 저장소에 저장됩니다. 따라서 Trident를 위한 별도의 저장소가 필요하지 않습니다.

19.07 릴리즈부터 Trident는 다양한 버전을 사용합니다 CustomResourceDefinition Trident 백 엔드, Trident 스토리지 클래스, Trident 볼륨과 같은 Trident 개체의 ID를 보존할 개체입니다. 이러한 오브젝트는 Trident에서 관리합니다. 또한 CSI 볼륨 스냅샷 프레임워크는 볼륨 스냅샷을 정의하는 데 필요한 일부 CRD를 소개합니다.

CRD는 Kubernetes를 구성하는 것입니다. 위에 정의된 리소스의 객체는 Trident에 의해 생성됩니다. 간단한 예로, 를 사용하여 백엔드를 생성할 수 있습니다 tridentctl, 해당 tridentbackends CRD 객체는 Kubernetes에서 사용할 수 있도록 생성되었습니다.

다음은 Trident의 CRD에 대해 고려해야 할 몇 가지 사항입니다.

- Trident가 설치되면 일련의 CRD가 생성되어 다른 리소스 유형과 마찬가지로 사용할 수 있습니다.
- Trident의 이전 버전(사용된 버전)에서 업그레이드할 때 `etcd` 상태를 유지하기 위해), Trident 설치 프로그램이 에서 데이터를 마이그레이션합니다 `etcd` 키 값 데이터 저장소 및 해당 CRD 개체 생성
- 를 사용하여 Trident를 제거하는 경우 `tridentctl uninstall` Command, Trident Pod가 삭제되지만 생성된 CRD는 정리되지 않습니다. 을 참조하십시오 "Trident를 제거합니다" Trident를 완전히 제거하고 처음부터 다시 구성할 수 있는 방법을 이해합니다.

트라이던트 `StorageClass` 오브젝트

Trident가 Kubernetes에 맞는 스토리지 클래스를 생성합니다 `StorageClass` 지정하는 개체입니다 `csi.trident.netapp.io/netapp.io/trident` 그들의 공급자 분야. 스토리지 클래스 이름이 Kubernetes의 클래스 이름과 일치합니다 `StorageClass` 나타내는 개체입니다.

> Kubernetes를 사용하면 이러한 오브젝트는 Kubernetes에서 자동으로 생성됩니다 `StorageClass` Trident를 프로비저닝자로 사용하는 등록이 완료되었습니다.

스토리지 클래스는 볼륨에 대한 일련의 요구 사항으로 구성됩니다. Trident는 이러한 요구 사항을 각 스토리지 풀에 있는 속성과 일치시킵니다. 일치하는 경우 해당 스토리지 풀이 해당 스토리지 클래스를 사용하여 볼륨을 프로비저닝할 수 있는 유효한 타겟입니다.

REST API를 사용하여 스토리지 클래스를 직접 정의하는 스토리지 클래스 구성을 생성할 수 있습니다. 그러나 Kubernetes 배포의 경우 새 Kubernetes 등록 시 Kubernetes가 생성될 것으로 예상합니다 `StorageClass` 오브젝트.

Trident 백엔드 객체

백엔드는 Trident가 볼륨을 프로비저닝하는 스토리지 공급자를 나타냅니다. 단일 Trident 인스턴스가 원하는 수의 백엔드를 관리할 수 있습니다.

> 이것은 직접 만들고 관리하는 두 가지 개체 유형 중 하나입니다. 다른 하나는 Kubernetes입니다 `StorageClass` 오브젝트.

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

트라이던트 `StoragePool` 오브젝트

스토리지 풀은 각 백엔드에서 용량 할당에 사용할 수 있는 고유한 위치를 나타냅니다. ONTAP의 경우 SVM에 있는 애그리게이트와 대응합니다. NetApp HCI/SolidFire의 경우 관리자 지정 QoS 밴드에 해당합니다. Cloud Volumes Service의 경우 클라우드 공급자 지역에 해당합니다. 각 스토리지 풀에는 고유한 스토리지 특성 세트가 있으며, 이 특성 집합은 성능 특성과 데이터 보호 특성을 정의합니다.

다른 오브젝트와 달리 스토리지 풀 후보 는 항상 자동으로 검색되고 관리됩니다.

트라이던트 `Volume` 오브젝트

볼륨은 NFS 공유 및 iSCSI LUN과 같은 백엔드 엔드포인트로 구성된 기본 프로비저닝 단위입니다. Kubernetes에서 이러한 항목은 에 직접 대응합니다 `PersistentVolumes`. 볼륨을 생성할 때 볼륨의 용량을 할당할 수 있는 위치와 크기를 결정하는 스토리지 클래스가 있는지 확인합니다.

Kubernetes에서 이러한 오브젝트는 자동으로 관리됩니다. 프로비저닝 Trident를 보려면 해당 Trident를 확인하십시오.

연결된 스냅샷이 있는 PV를 삭제하면 해당 Trident 볼륨이 * Deleting * 상태로 업데이트됩니다. Trident 볼륨을 삭제하려면 볼륨의 스냅샷을 제거해야 합니다.

볼륨 구성은 프로비저닝된 볼륨에 있어야 하는 속성을 정의합니다.

속성	유형	필수 요소입니다	설명
버전	문자열	아니요	Trident API 버전("1")
이름	문자열	예	생성할 볼륨의 이름입니다
storageClass 를 선택합니다	문자열	예	볼륨을 프로비저닝할 때 사용할 스토리지 클래스입니다
크기	문자열	예	용량 할당할 볼륨의 크기(바이트)입니다
프로토콜	문자열	아니요	사용할 프로토콜 유형;"파일" 또는 "블록"
내부 이름	문자열	아니요	스토리지 시스템에 있는 객체의 이름으로, Trident에서 생성
CloneSourceVolume	문자열	아니요	ONTAP(NAS, SAN) 및 SolidFire - *: 복제할 볼륨의 이름입니다
SplitOnClone 을 참조하십시오	문자열	아니요	ONTAP(NAS, SAN): 상위 클론에서 클론을 분할합니다
스냅샷 정책	문자열	아니요	ONTAP - *: 사용할 스냅샷 정책
snapshotReserve	문자열	아니요	ONTAP - *: 스냅숏용으로 예약된 볼륨의 비율입니다
내보내기 정책	문자열	아니요	ONTAP-NAS *: 사용할 엑스포트 정책
스냅샷 디렉토리	불입니다	아니요	ONTAP-NAS *: 스냅샷 디렉토리가 표시되는지 여부를 나타냅니다
unixPermissions	문자열	아니요	ONTAP-NAS *: 초기 UNIX 권한
블록 크기	문자열	아니요	SolidFire - *: 블록/섹터 크기
파일 시스템	문자열	아니요	파일 시스템 유형입니다

Trident가 생성합니다 `internalName` 볼륨을 생성할 때 이 단계는 두 단계로 구성됩니다. 먼저, 스토리지 접두어 앞에 추가됩니다(기본값 중 하나) `trident` 또는 백엔드 구성의 접두사)를 볼륨 이름에 입력하여 양식 이름을 만듭니다 `<prefix>-<volume-name>`. 그런 다음 백엔드에서 허용되지 않는 문자를 대체하여 이름을 삭제하는 작업을

진행합니다. ONTAP 백엔드의 경우 하이픈을 밑줄로 대체하므로 내부 이름은 이 됩니다 <prefix>_<volume-name>)를 클릭합니다. 요소 백엔드의 경우 밑줄을 하이픈으로 바꿉니다.

볼륨 구성을 사용하여 REST API를 사용하여 볼륨을 직접 프로비저닝할 수 있지만 Kubernetes 배포에서는 대부분의 사용자가 표준 Kubernetes를 사용할 것으로 예상합니다 PersistentVolumeClaim 방법. Trident는 프로비저닝 프로세스의 일부로 이 볼륨 개체를 자동으로 만듭니다.

트라이던트 Snapshot 오브젝트

스냅샷은 볼륨의 시점 복제본으로, 새 볼륨을 용량 할당하거나 복구 상태를 복구하는 데 사용할 수 있습니다. Kubernetes에서 이러한 항목은 에 직접 대응합니다 VolumeSnapshotContent 오브젝트. 각 스냅샷은 스냅샷에 대한 데이터의 소스인 볼륨에 연결됩니다.

각각 Snapshot 개체에는 아래 나열된 속성이 포함됩니다.

속성	유형	필수 요소입니다	설명
버전	문자열	예	Trident API 버전("1")
이름	문자열	예	Trident 스냅샷 개체의 이름입니다
내부 이름	문자열	예	스토리지 시스템의 Trident 스냅샷 개체의 이름입니다
볼륨 이름	문자열	예	스냅샷이 생성된 영구 볼륨의 이름입니다
볼륨 국제 이름	문자열	예	스토리지 시스템에서 연결된 Trident 볼륨 개체의 이름입니다

Kubernetes에서 이러한 오브젝트는 자동으로 관리됩니다. 프로비저닝 Trident를 보려면 해당 Trident를 확인하십시오.

Kubernetes를 사용할 경우 VolumeSnapshot 객체 요청이 생성되면 Trident는 백업 스토리지 시스템에 스냅샷 객체를 생성하여 작동합니다. 를 클릭합니다 internalName 이 스냅샷 개체의 접두어를 결합하여 생성됩니다 snapshot- 를 사용하여 UID 의 VolumeSnapshot 개체(예: snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660)를 클릭합니다. volumeName 및 volumeInternalName 백업 볼륨의 세부 정보를 가져오는 방식으로 채워집니다.

아스트라 트리덴트 ResourceQuota 오브젝트

Trident deamonset은 을 사용합니다 system-node-critical 우선 순위 클래스 - Kubernetes에서 가장 높은 우선 순위 클래스 - Astra Trident가 정상 노드 종료 중에 볼륨을 식별 및 정리하고, Trident에서 POD를 사용하여 리소스 압력이 높은 클러스터에서 낮은 우선 순위로 워크로드를 사전 예방할 수 있습니다.

이를 위해 Astra Trident는 을(를) 사용합니다 ResourceQuota Trident 데모에서 "system-node-critical" 우선 순위 클래스가 만족되는지 확인하는 개체입니다. Astra Trident는 구축 및 디멘션 생성 전에 을(를) 찾습니다 ResourceQuota 객체를 검색한 후, 검색되지 않은 경우 적용합니다.

기본 리소스 할당량 및 우선순위 클래스에 대한 추가 제어가 필요한 경우 을 생성할 수 있습니다 custom.yaml 또는 를 구성합니다 ResourceQuota 제어 차트를 사용하는 개체.

다음은 Trident 데모의 우선 순위를 지정하는 'ResourceQuota' 개체의 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 을 참조하십시오 "Kubernetes: 리소스 할당량".

정리 `ResourceQuota` 설치에 실패한 경우

드문 경우지만 이후 설치가 실패하는 경우가 있습니다 `ResourceQuota` 객체가 생성되었습니다. 먼저 시도하십시오 "제거 중" 그런 다음 다시 설치합니다.

이 기능이 작동하지 않으면 를 수동으로 제거합니다 `ResourceQuota` 오브젝트.

제거 `ResourceQuota`

자체 리소스 할당을 제어하려는 경우 Astra Trident를 제거할 수 있습니다 `ResourceQuota` 다음 명령을 사용하는 개체:

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

tridentctl 명령 및 옵션

를 클릭합니다 "Trident 설치 프로그램 번들" 명령줄 유틸리티, `tridentctl`Astra Trident에 간편하게 액세스할 수 있습니다. 충분한 권한을 가진 Kubernetes 사용자는 이를 사용하여 Astra Trident를 설치할 뿐 아니라 직접 상호 작용하여 Astra Trident Pod가 포함된 네임스페이스를 관리할 수 있습니다.

사용 가능한 명령 및 옵션

사용 정보를 보려면 를 실행합니다 `tridentctl --help`.

사용 가능한 명령 및 전역 옵션은 다음과 같습니다.

```
Usage:
  tridentctl [command]
```

사용 가능한 명령:

- `create`Astra Trident에 리소스를 추가합니다.
- `delete`Astra Trident에서 하나 이상의 리소스를 제거합니다.
- `get`Astra Trident에서 하나 이상의 리소스를 제공합니다.
- `help`: 명령에 대한 도움말입니다.
- `images`Astra Trident에 필요한 컨테이너 이미지 테이블을 인쇄합니다.
- `import`Astra Trident로 기존 리소스를 가져옵니다.
- `install`Astra Trident를 설치합니다.
- `logs`Astra Trident에서 로그를 인쇄합니다.
- `send`Astra Trident에서 리소스를 보냅니다.
- `uninstall`Astra Trident를 제거합니다.
- `update`Astra Trident에서 리소스를 수정합니다.
- `upgrade`: Astra Trident에서 리소스를 업그레이드합니다.
- `version`Astra Trident의 버전을 인쇄하십시오.

플래그:

- `-d, --debug`: 출력 디버그.
- `-h, --help`도움말을 참조하십시오 `tridentctl`.
- `-n, --namespace string`: Astra Trident 배포의 네임스페이스입니다.
- ``-o, --output string`출력 형식. json|YAML|name|wide|ps(기본값) 중 하나.
- `-s, --server string`: Astra Trident REST 인터페이스의 주소/포트입니다.

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

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

create

를 실행할 수 있습니다 `create` Astra Trident에 리소스를 추가하는 명령입니다.

```
Usage:
  tridentctl create [option]
```

사용 가능한 옵션:
`backend`Astra Trident에 백엔드를 추가합니다.

delete

를 실행할 수 있습니다 `delete` Astra Trident에서 하나 이상의 리소스를 제거하는 명령입니다.

```
Usage:
  tridentctl delete [option]
```

사용 가능한 옵션:

- `backend`Astra Trident에서 하나 이상의 스토리지 백엔드를 삭제합니다.
- `snapshot`Astra Trident에서 하나 이상의 볼륨 스냅샷을 삭제합니다.
- `storageclass`Astra Trident에서 하나 이상의 스토리지 클래스를 삭제합니다.
- `volume`Astra Trident에서 하나 이상의 스토리지 볼륨을 삭제합니다.

get

를 실행할 수 있습니다 `get` Astra Trident에서 하나 이상의 리소스를 제공하는 명령입니다.

```
Usage:
  tridentctl get [option]
```

사용 가능한 옵션:

- `backend`Astra Trident에서 하나 이상의 스토리지 백엔드를 받으세요.
- `snapshot`: Astra Trident에서 하나 이상의 스냅샷을 가져옵니다.
- `storageclass`Astra Trident에서 하나 이상의 스토리지 클래스를 다운로드하십시오.
- `volume`Astra Trident에서 하나 이상의 볼륨을 가져오십시오.

`volume` 플래그: * `-h, --help`: 볼륨에 대한 도움말입니다. * `--parentOfSubordinate string`: 하위 원본 볼륨으로 쿼리를 제한합니다. * `--subordinateOf string`: 볼륨 부하로 쿼리 제한.

images

를 실행할 수 있습니다 `images` Astra Trident에 필요한 컨테이너 이미지 테이블을 인쇄하려면 플래그를 지정합니다.

```
Usage:
  tridentctl images [flags]
```

플래그: * `-h, --help`: Help for images.
* -v, - -k8s -version string ': Kubernetes 클러스터의 의미 있는 버전

import volume

를 실행할 수 있습니다 `import volume` 기존 볼륨을 Astra Trident로 가져오는 명령입니다.

```
Usage:
  tridentctl import volume <backendName> <volumeName> [flags]
```

별칭:
`volume, v`

플래그:

- ``-f, --filename string`YAML 또는 JSON PVC 파일로 이동합니다.
- `-h, --help`: 볼륨에 대한 도움말입니다.
- `--no-manage`:PV/PVC만 생성 볼륨 라이프사이클 관리를 가정하지 마십시오.

install

를 실행할 수 있습니다 `install` Astra Trident를 설치하는 플래그입니다.

```
Usage:
  tridentctl install [flags]
```

플래그:

- `--autosupport-image string`: AutoSupport 원격 측정(기본값: "NetApp/트리덴트 자동 지원: 20.07.0")의 컨테이너 이미지입니다.
- `--autosupport-proxy string`: AutoSupport 텔레메트리 전송을 위한 프록시의 주소/포트입니다.
- ``--csi`CSI Trident 설치(Kubernetes 1.13에만 재정의, 기능 게이트 필요)
- `--enable-node-prep`: 노드에 필요한 패키지 설치를 시도합니다.
- `--generate-custom-yaml`: 아무 것도 설치하지 않고 YAML 파일을 생성합니다.
- `-h, --help`: 설치 도움말.
- `--http-request-timeout`: Trident 컨트롤러의 REST API에 대한 HTTP 요청 시간 초과를 재정의합니다 (기본값 1m30s).
- `--image-registry string`: 내부 이미지 레지스트리의 주소/포트입니다.

- ``--k8s-timeout duration`모든 Kubernetes 작업(기본값 3m0의)의 시간 초과.
- `--kubelet-dir string`: kubelet의 내부 상태(기본값 "/var/lib/kubelet")의 호스트 위치입니다.
- ``--log-format string`Astra Trident 로깅 형식(text, json)(기본 "text").
- ``--pv string`Astra Trident에서 사용하는 레거시 PV의 이름입니다. 이 이름이 존재하지 않는지 확인합니다(기본 "삼중류").
- ``--pvc string`Astra Trident에서 사용하는 기존 PVC의 이름입니다. 이 이름이 존재하지 않는지 확인합니다(기본 "삼중류").
- ``--silence-autosupport`AutoSupport 번들을 NetApp에 자동으로 보내지 않습니다(기본값: true).
- `--silent`: 설치하는 동안 대부분의 출력을 비활성화합니다.
- `--trident-image string`: 설치할 Astra Trident 이미지.
- ``--use-custom-yaml`설정 디렉토리에 있는 기존 YAML 파일을 사용합니다.
- `--use-ipv6`: Astra Trident의 통신에는 IPv6를 사용합니다.

logs

를 실행할 수 있습니다 `logs` Astra Trident의 로그를 인쇄할 플래그입니다.

```
Usage:
  tridentctl logs [flags]
```

플래그:

- `-a, --archive`: 별도로 지정하지 않는 한 모든 로그를 사용하여 지원 아카이브를 생성합니다.
- `-h, --help`: 로그 도움말.
- `-l, --log string`: 표시할 Astra Trident 로그. 트리덴트|auto|트리덴트-operator|all 중 하나(기본 "자동").
- ``--node string`노드 POD 로그를 수집할 Kubernetes 노드 이름입니다.
- `-p, --previous`: 이전 컨테이너 인스턴스에 대한 로그가 있으면 가져옵니다.
- `--sidecars`: 사이드카 컨테이너의 로그를 가져옵니다.

send

를 실행할 수 있습니다 `send` Astra Trident에서 리소스를 보내는 명령입니다.

```
Usage:
  tridentctl send [option]
```

사용 가능한 옵션:
`autosupport`AutoSupport 아카이브를 NetApp으로 전송합니다.

uninstall

를 실행할 수 있습니다 `uninstall` Astra Trident를 제거하는 플래그입니다.

```
Usage:
  tridentctl uninstall [flags]
```

플래그: * `-h, --help`: 제거 도움말입니다. * `--silent`: 제거 중 대부분의 출력을 비활성화합니다.

update

를 실행할 수 있습니다 `update` Astra Trident에서 리소스를 수정하는 명령입니다.

```
Usage:
  tridentctl update [option]
```

사용 가능한 옵션:
`backend`Astra Trident에서 백엔드를 업데이트합니다.

upgrade

를 실행할 수 있습니다 `upgrade` Astra Trident에서 리소스를 업그레이드하는 명령입니다.

```
Usage:
tridentctl upgrade [option]
```

사용 가능한 옵션:
`volume`: NFS/iSCSI에서 CSI로 하나 이상의 영구 볼륨을 업그레이드합니다.

version

를 실행할 수 있습니다 `version` 플래그를 사용하여 의 버전을 인쇄합니다 `tridentctl` 및 실행 중인 Trident 서비스를 제공합니다.

```
Usage:
  tridentctl version [flags]
```

플래그: * `--client`: 클라이언트 버전만(서버가 필요하지 않음). * `-h, --help`: 버전에 대한 도움말입니다.

POD 보안 표준(PSS) 및 보안 컨텍스트 제약(SCC)

Kubernetes Pod 보안 표준(PSS) 및 Pod 보안 정책(PSP)에서 사용 권한 수준을 정의하고

Pod의 동작을 제한합니다. OpenShift Security Context Constraints(SCC)도 OpenShift Kubernetes Engine에 특정한 POD 제한을 정의합니다. 이러한 사용자 지정을 제공하기 위해 Astra Trident는 설치 중에 특정 권한을 활성화합니다. 다음 섹션에서는 Astra Trident에서 설정한 사용 권한에 대해 자세히 설명합니다.

PSS는 Pod 보안 정책(PSP)을 대체합니다. PSP는 Kubernetes v1.21에서 사용되지 않으며 v1.25에서 제거됩니다. 자세한 내용은 을 참조하십시오 "Kubernetes: 보안".

필요한 **Kubernetes** 보안 컨텍스트 및 관련 필드

권한	설명
특별 권한	CSI를 사용하려면 마운트 지점이 양방향이어야 합니다. 즉, Trident 노드 포드가 권한이 있는 컨테이너를 실행해야 합니다. 자세한 내용은 을 참조하십시오 "Kubernetes: 마운트 전파".
호스트 네트워킹	iSCSI 데몬에 필요합니다. `iscsiadm` iSCSI 마운트를 관리하고 호스트 네트워킹을 사용하여 iSCSI 데몬과 통신합니다.
호스트 IPC	NFS는 IPC(프로세스 간 통신)를 사용하여 NFSD와 통신합니다.
호스트 PID	를 시작해야 합니다 `rpc-statd` 대해 NFS를 선택합니다. Astra Trident가 호스트 프로세스를 쿼리하여 확인 여부를 결정합니다 `rpc-statd` NFS 볼륨을 마운트하기 전에 실행 중입니다.

제공합니다	를 클릭합니다 `SYS_ADMIN` 권한이 있는 컨테이너에 대한 기본 기능의 일부로 기능이 제공됩니다. 예를 들어, Docker는 권한이 있는 컨테이너에 대해 다음 기능을 설정합니다.
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

Seccomp	Seccomp 프로필은 권한 있는 컨테이너에서 항상 "제한 없음"이므로 Astra Trident에서 활성화할 수 없습니다.
SELinux	OpenShift에서는 권한이 있는 컨테이너가 에서 실행됩니다 `spc_t` ("슈퍼 프리권한 컨테이너") 도메인 및 권한 없는 컨테이너는 에서 실행됩니다 `container_t` 도메인. 켜짐 `containerd`, 및 `container-selinux` 설치된 모든 컨테이너는 에서 실행됩니다 `spc_t` SELinux를 효과적으로 비활성화하는 도메인입니다. 따라서 Astra Trident는 추가되지 않습니다 `seLinuxOptions` 컨테이너로.
DAC	권한이 있는 컨테이너는 루트로 실행되어야 합니다. 권한이 없는 컨테이너는 root로 실행되어 CSI에 필요한 UNIX 소켓에 액세스합니다.

POD 보안 표준(PSS)

라벨	설명	기본값

`pod-security.kubernetes.io/enforce`
 	
`pod-security.kubernetes.io/enforce-version`
	Trident 컨트롤러와 노드를 설치 네임스페이스에 받아들일 수 있습니다. 네임스페이스 레이블을 변경하지 마십시오.	`enforce: privileged`
		
		`enforce-version: <version of the current cluster or highest version of PSS tested.>`

네임스페이스 레이블을 변경하면 포드가 예약되지 않고 "오류 생성:..." 또는 "경고: 트리덴트 - CSI -..." 가 발생할 수 있습니다. 이 경우 에 대한 네임스페이스 레이블이 있는지 확인합니다 `privileged` 변경되었습니다. 있는 경우 Trident를 다시 설치합니다.

PSP(POD 보안 정책)

필드에 입력합니다	설명	기본값
`allowPrivilegeEscalation`	권한 있는 컨테이너는 권한 에스컬레이션을 허용해야 합니다.	`true`
`allowedCSIDrivers`	Trident는 인라인 CSI 임시 볼륨을 사용하지 않습니다.	비어 있습니다
`allowedCapabilities`	권한이 없는 Trident 컨테이너는 기본 세트보다 더 많은 기능을 필요로 하지 않으며 권한이 있는 컨테이너에 모든 가능한 기능이 부여됩니다.	비어 있습니다
`allowedFlexVolumes`	Trident는 을 사용하지 않습니다 "FlexVolume 드라이버"따라서 허용된 볼륨 목록에 포함되지 않습니다.	비어 있습니다
`allowedHostPaths`	Trident 노드 포드는 노드의 루트 파일 시스템을 마운트하므로 이 목록을 설정하는 데는 아무런 이점이 없습니다.	비어 있습니다
`allowedProcMountTypes`	Trident는 아무 용하지 않습니다 `ProcMountTypes`.	비어 있습니다
`allowedUnsafeSysctls`	Trident는 안전하지 않을 필요가 없습니다 `sysctls`.	비어 있습니다
`defaultAddCapabilities`	권한이 있는 컨테이너에 기능을 추가할 필요가 없습니다.	비어 있습니다
`defaultAllowPrivilegeEscalation`	권한 에스컬레이션을 허용하는 작업은 각 Trident 포드에서 처리됩니다.	`false`
`forbiddenSysctls`	아니요 `sysctls` 허용됩니다.	비어 있습니다
`fsGroup`	Trident 컨테이너가 루트로 실행됩니다.	`RunAsAny`

필드에 입력합니다	설명	기본값
`hostIPC`	NFS 볼륨을 마운트하려면 호스트 IPC가 와 통신해야 합니다 `nfsd`	`true`
`hostNetwork`	iscsiadm을 사용하려면 호스트 네트워크가 iSCSI 데몬과 통신해야 합니다.	`true`
`hostPID`	호스트 PID가 있는지 확인해야 합니다 `rpc-statd` 노드에서 실행 중입니다.	`true`
`hostPorts`	Trident는 호스트 포트를 사용하지 않습니다.	비어 있습니다
`privileged`	Trident 노드 포드는 볼륨을 마운트하려면 권한이 있는 컨테이너를 실행해야 합니다.	`true`
`readOnlyRootFilesystem`	Trident 노드 포드는 노드 파일 시스템에 써야 합니다.	`false`
`requiredDropCapabilities`	Trident 노드 포드는 권한이 있는 컨테이너를 실행하고 기능을 삭제할 수 없습니다.	`none`
`runAsGroup`	Trident 컨테이너가 루트로 실행됩니다.	`RunAsAny`
`runAsUser`	Trident 컨테이너가 루트로 실행됩니다.	`runAsAny`
`runtimeClass`	Trident가 사용되지 않습니다 `RuntimeClasses`.	비어 있습니다
`seLinux`	Trident가 설정되지 않았습니다 `seLinuxOptions` 현재 컨테이너 실행 시간과 Kubernetes 배포에서 SELinux를 처리하는 방법은 서로 다릅니다.	비어 있습니다
`supplementalGroups`	Trident 컨테이너가 루트로 실행됩니다.	`RunAsAny`
`volumes`	Trident Pod에는 이러한 볼륨 플러그인이 필요합니다.	`hostPath, projected, emptyDir`

SCC(Security Context Constraints)

라벨	설명	기본값
`allowHostDirVolumePlugin`	Trident 노드 포드는 노드의 루트 파일 시스템을 마운트합니다.	`true`
`allowHostIPC`	NFS 볼륨을 마운트하려면 호스트 IPC가 와 통신해야 합니다 `nfsd`.	`true`

라벨	설명	기본값
`allowHostNetwork`	iscsiadm을 사용하려면 호스트 네트워크가 iSCSI 데몬과 통신해야 합니다.	`true`
`allowHostPID`	호스트 PID가 있는지 확인해야 합니다 `rpc-statd` 노드에서 실행 중입니다.	`true`
`allowHostPorts`	Trident는 호스트 포트를 사용하지 않습니다.	`false`
`allowPrivilegeEscalation`	권한 있는 컨테이너는 권한 에스컬레이션을 허용해야 합니다.	`true`
`allowPrivilegedContainer`	Trident 노드 포드는 볼륨을 마운트하려면 권한이 있는 컨테이너를 실행해야 합니다.	`true`
`allowedUnsafeSysctls`	Trident는 안전하지 않을 필요가 없습니다 `sysctls`.	`none`
`allowedCapabilities`	권한이 없는 Trident 컨테이너는 기본 세트보다 더 많은 기능을 필요로 하지 않으며 권한이 있는 컨테이너에 모든 가능한 기능이 부여됩니다.	비어 있습니다
`defaultAddCapabilities`	권한이 있는 컨테이너에 기능을 추가할 필요가 없습니다.	비어 있습니다
`fsGroup`	Trident 컨테이너가 루트로 실행됩니다.	`RunAsAny`
`groups`	이 SCC는 Trident에만 해당되며 사용자에게 바인딩됩니다.	비어 있습니다
`readOnlyRootFilesystem`	Trident 노드 포드는 노드 파일 시스템에 써야 합니다.	`false`
`requiredDropCapabilities`	Trident 노드 포드는 권한이 있는 컨테이너를 실행하고 기능을 삭제할 수 없습니다.	`none`
`runAsUser`	Trident 컨테이너가 루트로 실행됩니다.	`RunAsAny`
`seLinuxContext`	Trident가 설정되지 않았습니다 `seLinuxOptions` 현재 컨테이너 실행 시간과 Kubernetes 배포에서 SELinux를 처리하는 방법은 서로 다릅니다.	비어 있습니다
`seccompProfiles`	특권 컨테이너는 항상 "비제한" 상태로 실행됩니다.	비어 있습니다
`supplementalGroups`	Trident 컨테이너가 루트로 실행됩니다.	`RunAsAny`
`users`	이 SCC를 Trident 네임스페이스의 Trident 사용자에게 바인딩하기 위해 하나의 항목이 제공됩니다.	해당 없음

라벨	설명	기본값
`volumes`	Trident Pod에는 이러한 볼륨 플러그인이 필요합니다.	`hostPath, downwardAPI, projected, emptyDir`