시작하십시오
Astra Trident

NetApp
April 16, 2024

목차

시작하십시오

시도해 보십시오

NetApp은 사용자가 요청할 수 있는 즉시 사용 가능한 연구소 이미지를 제공합니다 "NetApp 시험 구동".

시험 구동에 대해 자세히 알아보십시오

테스트 드라이브는 3노드 Kubernetes 클러스터 및 Astra Trident가 설치 및 구성된 샌드박스 환경을 제공합니다. Astra Trident에 대해 알아보고 기능을 둘러보는 것도 좋습니다.

또 다른 옵션은 을 보는 것입니다 "kubeadm 설치 가이드" Kubernetes에서 제공:

> 이러한 지침을 운영 환경에 사용하여 구축한 Kubernetes 클러스터를 사용해서는 안 됩니다. 배포 시 제공되는 운영 구축 가이드를 사용하여 즉시 프로덕션할 수 있는 클러스터를 생성할 수 있습니다.

Kubernetes를 처음 사용하는 경우 개념 및 툴에 대해 자세히 알아보십시오 "여기".

요구 사항

지원되는 프런트엔드, 백엔드 및 호스트 구성을 검토하여 시작하십시오.

> Astra Trident가 사용하는 포트에 대한 자세한 내용은 를 참조하십시오 "여기".

Astra Trident 22.10에 대한 중요 정보입니다

- Astra Trident 22.10으로 업그레이드하기 전에 다음 중요 정보를 읽어야 합니다. *

> **strong>Astra Trident 22.10**에 대한 중요 정보****
>
> - 이제 Trident에서 Kubernetes 1.25가 지원됩니다. Kubernetes 1.25로 업그레이드하기 전에 Astra Trident 22.10으로 업그레이드해야 합니다.
> - Astra Trident는 이제 SAN 환경에서 다중 경로 구성을 사용하도록 엄격히 적용되며 권장값은 입니다 `find_multipaths: no` 다중 경로 .conf 파일
>
> 비 경로 다중화 구성 또는 의 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트 오류가 발생합니다. Trident에서 의 사용을 권장했습니다 `find_multipaths: no` 21.07 릴리스 이후.

지원되는 프런트엔드(오케스트레이터)

Astra Trident는 다음을 비롯한 여러 컨테이너 엔진과 오케스트레이터가 지원됩니다.

- Anthos On-Premise(VMware) 및 Anthos의 Bare Metal 1.9, 1.10, 1.11
- Kubernetes 1.20-1.25

- Mirantis Kubernetes 엔진 3.5
- OpenShift 4.8, 4.9, 4.10, 4.11

Trident 연산자는 다음 릴리즈에서 지원됩니다.

- Anthos On-Premise(VMware) 및 Anthos의 Bare Metal 1.9, 1.10, 1.11
- Kubernetes 1.20-1.25
- OpenShift 4.8, 4.9, 4.10, 4.11

Astra Trident는 GKE(Google Kubernetes Engine), EKS(Amazon Elastic Kubernetes Service), AKS(Azure Kubernetes Service), Rancher, VMware Tanzu Portfolio를 비롯한 기타 완전 관리형 및 자체 관리 Kubernetes 오퍼링과도 작동합니다.

지원되는 백엔드(스토리지)

Astra Trident를 사용하려면 다음 중 하나 이상의 지원되는 백엔드가 필요합니다.

- NetApp ONTAP용 Amazon FSx
- Azure NetApp Files
- Cloud Volumes ONTAP
- GCP용 Cloud Volumes Service
- FAS/AFF/9.3 이상을 선택합니다
- NetApp All SAN 어레이(ASA)
- NetApp HCI/Element 소프트웨어 11 이상

피처 요구 사항

아래 표에는 Astra Trident의 이번 릴리즈와 함께 사용할 수 있는 기능과 지원하는 Kubernetes 버전이 요약되어 있습니다.

피처	**Kubernetes** 버전	기능 게이트가 필요합니까**?**
CSI Trident	1.20-1.25	아니요
볼륨 스냅샷	1.20-1.25	아니요
체적 스냅샷의 PVC	1.20-1.25	아니요
iSCSI PV 크기 조정	1.20-1.25	아니요
ONTAP 양방향 CHAP	1.20-1.25	아니요
동적 내보내기 정책	1.20-1.25	아니요
Trident 연산자	1.20-1.25	아니요

피처	**Kubernetes** 버전	기능 게이트가 필요합니까**?**
Auto Worker Node Prep(베타)	1.20-1.25	아니요
CSI 토폴로지	1.20-1.25	아니요

호스트 운영 체제를 테스트했습니다

Astra Trident는 특정 운영 체제를 공식적으로 "지원"하지 않지만 다음과 같은 운영 체제가 작동하는 것으로 알려져 있습니다.

- OpenShift Container Platform에서 지원하는 RedHat CoreOS(RHCOS) 버전
- RHEL 또는 CentOS 7
- Ubuntu 18.04 이상(최신 22.04)
- Windows Server 2019

기본적으로 Astra Trident는 컨테이너에서 실행되므로 모든 Linux 작업자에서 실행됩니다. 그러나 이러한 작업자들은 표준 NFS 클라이언트 또는 iSCSI 이니시에이터를 사용하여 Astra Trident가 제공하는 볼륨을 사용 중인 백엔드에 따라 마운트할 수 있어야 합니다.

를 클릭합니다 `tridentctl` 이 유틸리티는 이러한 Linux 배포판에서도 실행됩니다.

호스트 구성

사용 중인 백엔드에 따라 NFS 및/또는 iSCSI 유틸리티를 클러스터의 모든 작업자에 설치해야 합니다. 을 참조하십시오 "여기" 를 참조하십시오.

스토리지 시스템 구성

Astra Trident는 백엔드 구성에서 사용하기 전에 스토리지 시스템을 일부 변경해야 할 수 있습니다. 을 참조하십시오 "여기" 를 참조하십시오.

컨테이너 이미지 및 해당 **Kubernetes** 버전

공기 박형 설치의 경우 다음 목록은 Astra Trident를 설치하는 데 필요한 컨테이너 이미지의 참조입니다. 를 사용합니다 `tridentctl images` 명령을 사용하여 필요한 컨테이너 이미지 목록을 확인합니다.

Kubernetes 버전	컨테이너 이미지
v1.20.0	• Docker.IO/NetApp/트리덴트: 22.10.0 • Docker.IO/NetApp/트리덴트 - 자동 지원: 22.10 • registry.k8s.io/sig-storage/scsi-v3.3.3.0 • registry.k8s.io/SIG-storage/CSI-attacher:v4.0.0 • registry.k8s.io/SIG-storage/CSI-resizer: v1.6.0 • 레지스트리.k8s.io/sig-storage/csi-shotter:v6.1.0 • registry.k8s.io/SIG-storage/CSI-node-driver-registrar: v2.5.1 • Docker.IO/NetApp/트리덴트 - 운영자: 22.10.0(옵션)
v1.21.0	• Docker.IO/NetApp/트리덴트: 22.10.0 • Docker.IO/NetApp/트리덴트 - 자동 지원: 22.10 • registry.k8s.io/sig-storage/scsi-v3.3.3.0 • registry.k8s.io/SIG-storage/CSI-attacher:v4.0.0 • registry.k8s.io/SIG-storage/CSI-resizer: v1.6.0 • 레지스트리.k8s.io/sig-storage/csi-shotter:v6.1.0 • registry.k8s.io/SIG-storage/CSI-node-driver-registrar: v2.5.1 • Docker.IO/NetApp/트리덴트 - 운영자: 22.10.0(옵션)
v1.22.0	• Docker.IO/NetApp/트리덴트: 22.10.0 • Docker.IO/NetApp/트리덴트 - 자동 지원: 22.10 • registry.k8s.io/sig-storage/scsi-v3.3.3.0 • registry.k8s.io/SIG-storage/CSI-attacher:v4.0.0 • registry.k8s.io/SIG-storage/CSI-resizer: v1.6.0 • 레지스트리.k8s.io/sig-storage/csi-shotter:v6.1.0 • registry.k8s.io/SIG-storage/CSI-node-driver-registrar: v2.5.1 • Docker.IO/NetApp/트리덴트 - 운영자: 22.10.0(옵션)

Kubernetes 버전	컨테이너 이미지
v1.23.0	• Docker.IO/NetApp/트리덴트: 22.10.0 • Docker.IO/NetApp/트리덴트 - 자동 지원: 22.10 • registry.k8s.io/sig-storage/scsi-v3.3.3.0 • registry.k8s.io/SIG-storage/CSI-attacher:v4.0.0 • registry.k8s.io/SIG-storage/CSI-resizer: v1.6.0 • 레지스트리.k8s.io/sig-storage/csi-shotter:v6.1.0 • registry.k8s.io/SIG-storage/CSI-node-driver-registrar: v2.5.1 • Docker.IO/NetApp/트리덴트 - 운영자: 22.10.0(옵션)
v1.24.0	• Docker.IO/NetApp/트리덴트: 22.10.0 • Docker.IO/NetApp/트리덴트 - 자동 지원: 22.10 • registry.k8s.io/sig-storage/scsi-v3.3.3.0 • registry.k8s.io/SIG-storage/CSI-attacher:v4.0.0 • registry.k8s.io/SIG-storage/CSI-resizer: v1.6.0 • 레지스트리.k8s.io/sig-storage/csi-shotter:v6.1.0 • registry.k8s.io/SIG-storage/CSI-node-driver-registrar: v2.5.1 • Docker.IO/NetApp/트리덴트 - 운영자: 22.10.0(옵션)
v1.25.0	• Docker.IO/NetApp/트리덴트: 22.10.0 • Docker.IO/NetApp/트리덴트 - 자동 지원: 22.10 • registry.k8s.io/sig-storage/scsi-v3.3.3.0 • registry.k8s.io/SIG-storage/CSI-attacher:v4.0.0 • registry.k8s.io/SIG-storage/CSI-resizer: v1.6.0 • 레지스트리.k8s.io/sig-storage/csi-shotter:v6.1.0 • registry.k8s.io/SIG-storage/CSI-node-driver-registrar: v2.5.1 • Docker.IO/NetApp/트리덴트 - 운영자: 22.10.0(옵션)

Kubernetes 버전 1.20 이상에서 검증된 를 사용합니다 `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v6.x` ?곡길?? `v1` 에서 지원하는 버전입니다 `volumesnapshots.snapshot.storage.k8s.gcr.io` CRD 를 누릅니다 `v1beta1` 에서 CRD를 지원하는 버전입니다 `v1` 버전, 검증된 을 사용합니다 `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v3.x` 이미지.

구축 개요

Trident 운영자나 를 사용하여 Astra Trident를 배포할 수 있습니다 `tridentctl`.

22.04 릴리즈부터는 Astra Trident가 설치될 때마다 AES 키가 더 이상 다시 생성되지 않습니다. 이 릴리즈를 통해 Astra Trident는 설치 전반에 걸쳐 유지되는 새로운 비밀 객체를 설치합니다. 즉, `tridentctl` 22.04에서는 Trident의 이전 버전을 제거할 수 있지만 이전 버전에서는 22.04 설치를 제거할 수 없습니다.

Astra Trident 22.10에 대한 중요 정보입니다

- Astra Trident 22.10으로 업그레이드하기 전에 다음 중요 정보를 읽어야 합니다. *

strong>Astra Trident 22.10에 대한 중요 정보****

- 이제 Trident에서 Kubernetes 1.25가 지원됩니다. Kubernetes 1.25로 업그레이드하기 전에 Astra Trident 22.10으로 업그레이드해야 합니다.
- Astra Trident는 이제 SAN 환경에서 다중 경로 구성을 사용하도록 엄격히 적용되며 권장값은 입니다 `find_multipaths: no` 다중 경로 .conf 파일

  비 경로 다중화 구성 또는 의 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트 오류가 발생합니다. Trident에서 의 사용을 권장했습니다 `find_multipaths: no` 21.07 릴리스 이후.

배포 방법을 선택합니다

사용할 배포 방법을 결정하려면 다음 사항을 고려하십시오.

Trident 연산자를 사용하는 경우

를 클릭합니다 "Trident 운영자" Astra Trident 리소스를 동적으로 관리하고 설정 단계를 자동화하는 훌륭한 방법입니다. 몇 가지 전제 조건이 충족되어야 합니다. 을 참조하십시오 "설명합니다".

Trident 운영자는 아래에 설명된 여러 가지 이점을 제공합니다.

자동 복구 기능

Astra Trident 설치를 모니터링하고 구축이 삭제되거나 실수로 수정된 경우와 같은 문제를 해결하기 위한 조치를 적극적으로 취할 수 있습니다. 운용자가 배치로 설정된 경우, a `trident-operator-<generated-id>` POD가 생성됩니다. 이 포드는 을(를) 연결합니다 `TridentOrchestrator` Astra Trident가 설치된 CR을 사용하면 항상 한 번만 활성화되도록 할 수 있습니다 `TridentOrchestrator`. 즉, 운영자는 클러스터에 Astra Trident 인스턴스가 하나만 있고 설정을 제어하여 설치가 매우 강력한지 확인합니다. 설치 변경(예: 배포 또는 노드 반점 삭제)이 수행되면 운영자가 이를 식별하고 개별적으로 수정합니다.

기존 설치에 대한 손쉬운 업데이트

기존 배포를 운영자로 쉽게 업데이트할 수 있습니다. 를 편집하기만 하면 됩니다 `TridentOrchestrator` CR을 사용하여 설치를 업데이트합니다. 예를 들어, Astra Trident를 활성화하여 디버그 로그를 생성해야 하는 시나리오를 생각해 보십시오.

이렇게 하려면 에 패치를 적용합니다 `TridentOrchestrator` 를 눌러 설정합니다 `spec.debug` 를 선택합니다 `true`:

```
kubectl patch torc <trident-orchestrator-name> -n trident --type=merge -p
'{"spec":{"debug":true}}'
```

이후 `TridentOrchestrator` 이 업데이트되면 운영자가 업데이트를 처리하고 기존 설치를 패치합니다. 이렇게 하면 새 포드가 생성되어 설치를 적절하게 수정할 수 있습니다.

Kubernetes 업그레이드 자동 처리

클러스터의 Kubernetes 버전이 지원되는 버전으로 업그레이드되면 운영자는 기존 Astra Trident 설치를 자동으로 업데이트하고 Kubernetes 버전 요구사항을 충족하도록 변경합니다.

> ⓘ 클러스터가 지원되지 않는 버전으로 업그레이드되면 운영자는 Astra Trident를 설치할 수 없습니다. Astra Trident가 운영자와 함께 이미 설치된 경우 Astra Trident가 지원되지 않는 Kubernetes 버전에 설치되었음을 나타내는 경고가 표시됩니다.

BlueXP(이전의 Cloud Manager)를 사용하여 Kubernetes 클러스터 관리

와 함께 "BlueXP를 사용하는 Astra Trident"Astra Trident의 최신 버전으로 업그레이드하고, 스토리지 클래스를 추가 및 관리하고, 작업 환경에 연결한 다음, Cloud Backup Service를 사용하여 영구 볼륨을 백업할 수 있습니다. BlueXP는 Trident 연산자를 사용하여 수동으로 또는 Helm을 사용하여 Astra Trident 구축을 지원합니다.

제어 사용 시기

Astra Trident 21.01부터 Helm을 사용하여 관리하는 다른 애플리케이션이 있는 경우 Helm을 사용하여 배포를 관리할 수도 있습니다.

사용 시기 `tridenctl`

업그레이드해야 하는 기존 배포가 있거나 배포를 사용자 지정하려는 경우 을 사용하는 것이 좋습니다 "tridentctl 을 선택합니다". Astra Trident를 구축하는 기존 방법입니다.

배포 방법 간에 이동할 때의 고려 사항

배포 방법 간에 이동해야 하는 시나리오를 상상하기란 쉽지 않습니다. 에서 이동하기 전에 다음 사항을 고려해야 합니다 `tridentctl` 연산자 기반 배포로 배포하거나 그 반대로 배포:

- Astra Trident를 제거할 때는 항상 동일한 방법을 사용하십시오. 을(를) 배포한 경우 `tridentctl`, 의 해당 버전을 사용해야 합니다 `tridentctl` Astra Trident를 제거하는 바이너리. 마찬가지로 연산자를 사용하여 를 배포하는 경우에는 를 편집해야 합니다 `TridentOrchestrator` CR 및 SET `spec.uninstall=true` Astra Trident를 제거합니다.
- 를 제거하고 사용할 운영자 기반 배포가 있는 경우 `tridentctl` Astra Trident를 배포하려면 먼저 편집해야 합니다 `TridentOrchestrator` 그리고 설정합니다 `spec.uninstall=true` Astra Trident를 제거합니다. 그런 다음 삭제합니다 `TridentOrchestrator` 및 작업자 배포. 그런 다음 를 사용하여 를 설치할 수 있습니다 `tridentctl`.
- 작업자 기반의 수동 배포를 사용하고 H제어 기반 Trident 연산자 배포를 사용하려는 경우 먼저 수동으로 연산자를 제거한 다음 Helm 설치를 수행해야 합니다. 이를 통해 Helm은 필요한 레이블 및 주석을 사용하여 Trident 연산자를

배포할 수 있습니다. 이렇게 하지 않으면 레이블 유효성 검사 오류 및 주석 유효성 검사 오류와 함께 H제어 기반 Trident 연산자 배포가 실패합니다. 가 있는 경우 `tridentctl`기반 배포에서는 문제 없이 Helm 기반 배포를 사용할 수 있습니다.

배포 모드를 이해합니다

Astra Trident를 구축하는 방법에는 세 가지가 있습니다.

Standard 구축

Kubernetes 클러스터에서 Trident를 구축하면 Astra Trident 설치 관리자가 다음 두 가지 작업을 수행할 수 있습니다.

- 인터넷을 통해 컨테이너 이미지를 가져오는 중입니다
- Kubernetes 클러스터의 모든 유효한 노드에서 Astra Trident Pod를 가동하는 구축 및/또는 노드 데모 생성

이와 같은 표준 배포는 두 가지 방법으로 수행할 수 있습니다.

- 사용 `tridentctl install`
- Trident 연산자 사용 Trident 연산자는 수동으로 또는 Helm을 사용하여 배포할 수 있습니다.

이 설치 모드는 Astra Trident를 설치하는 가장 쉬운 방법이며 네트워크 제한이 없는 대부분의 환경에서 작동합니다.

오프라인 배포

공기 Gapped 배포를 수행하려면 를 사용합니다 `--image-registry` 호출 시 플래그 `tridentctl install` 개인 이미지 레지스트리를 가리킵니다. Trident 연산자를 사용하여 배포하는 경우 또는 을 지정할 수 있습니다 `spec.imageRegistry` 을 클릭합니다 `TridentOrchestrator`. 이 레지스트리에는 가 포함되어 있어야 합니다 "Trident 이미지", "Trident AutoSupport 이미지"및 Kubernetes 버전에서 요구하는 CSI 사이드카 이미지를 참조하십시오.

를 사용하여 배포를 사용자 지정할 수 있습니다 `tridentctl` Trident의 리소스에 대한 매니페스트를 생성합니다. 여기에는 구축, 개발/제거, 서비스 계정, Astra Trident가 설치의 일부로 생성한 클러스터 역할 등이 포함됩니다.

배포 사용자 지정에 대한 자세한 내용은 다음 링크를 참조하십시오.

- "운영자 기반 배포를 사용자 지정합니다"
*

> 개인 이미지 리포지토리를 사용하는 경우 를 추가해야 합니다 `/sig-storage` 개인 레지스트리 URL의 끝까지. 에 대한 개인 레지스트리를 사용하는 경우 `tridentctl` 배포는 을 사용해야 합니다 `--trident-image` 및 `--autosupport-image` 와 함께 제공됩니다 `--image-registry`. Trident 연산자를 사용하여 Astra Trident를 배포하는 경우 Orchestrator CR에 포함되어 있는지 확인합니다 `tridentImage` 및 `autosupportImage` 를 입력합니다.

원격 배포

다음은 원격 배포 프로세스에 대한 상위 수준의 개요입니다.

- 적절한 버전의 를 배포합니다 `kubectl` Astra Trident를 구축하려는 원격 머신
- Kubernetes 클러스터에서 구성 파일을 복사하고 를 설정합니다 `KUBECONFIG` 원격 시스템의 환경 변수.

- 를 시작합니다 `kubectl get nodes` 명령을 실행하여 필요한 Kubernetes 클러스터에 연결할 수 있는지 확인하십시오.
- 표준 설치 단계를 사용하여 원격 컴퓨터에서 배포를 완료합니다.

기타 알려진 구성 옵션

VMware Tanzu 포트폴리오 제품에 Astra Trident를 설치할 경우:

- 클러스터는 권한이 있는 워크로드를 지원해야 합니다.
- 를 클릭합니다 `--kubelet-dir` 플래그를 kubelet 디렉터리의 위치로 설정해야 합니다. 기본적으로 이 값은 입니다 `/var/vcap/data/kubelet`.

  를 사용하여 kubelet 위치 지정 `--kubelet-dir` Trident Operator, Helm 및 에 대해 작업하는 것으로 알려져 있습니다 `tridentctl` 적합합니다.

Trident 연산자를 사용하여 구축

Trident 연산자를 사용하여 Astra Trident를 배포할 수 있습니다.

Astra Trident 22.10에 대한 중요 정보입니다

- Astra Trident 22.10으로 업그레이드하기 전에 다음 중요 정보를 읽어야 합니다. *

strong>Astra Trident 22.10에 대한 중요 정보

- 이제 Trident에서 Kubernetes 1.25가 지원됩니다. Kubernetes 1.25로 업그레이드하기 전에 Astra Trident 22.10으로 업그레이드해야 합니다.
- Astra Trident는 이제 SAN 환경에서 다중 경로 구성을 사용하도록 엄격히 적용되며 권장값은 입니다 `find_multipaths: no` 다중 경로 .conf 파일

  비 경로 다중화 구성 또는 의 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트 오류가 발생합니다. Trident에서 의 사용을 권장했습니다 `find_multipaths: no` 21.07 릴리스 이후.

Trident 운영자 구축 옵션

Trident 연산자는 다음 두 가지 방법 중 하나로 배포할 수 있습니다.

- Trident 사용 "Helm 차트": 제어 도표는 Trident 연산자를 배포하고 Trident를 한 번에 설치합니다.
- 수동: Trident 연산자를 설치하고 관련 개체를 만드는 데 사용할 수 있는 파일을 제공합니다.
  - Kubernetes 1.24 이하 를 실행하는 클러스터의 경우, 를 사용합니다 "Bundle_PRE_1_25.YAML".
  - Kubernetes 1.25 이상을 실행하는 클러스터의 경우 를 사용합니다 "Bundle_post_1_25.YAML".

에 아직 익숙하지 않은 경우 "기본 개념"이제 아주 좋은 시간입니다.

필수 구성 요소를 확인합니다

Astra Trident를 구축하려면 다음과 같은 사전 요구 사항을 충족해야 합니다.

- 지원되는 Kubernetes 버전을 실행 중인 지원되는 Kubernetes 클러스터에 대한 모든 권한이 있습니다. 를 검토합니다 "요구 사항".
- 지원되는 NetApp 스토리지 시스템에 액세스할 수 있습니다.
- 모든 Kubernetes 작업자 노드에서 볼륨을 마운트할 수 있습니다.
- 에 Linux 호스트가 있습니다 `kubectl` (또는 `oc`, OpenShift를 사용하는 경우) 사용하려는 Kubernetes 클러스터를 관리하도록 설치 및 구성한 것입니다.
- 을(를) 설정했습니다 `KUBECONFIG` Kubernetes 클러스터 구성을 가리키는 환경 변수
- 을(를) 활성화했습니다 "Astra Trident에서 요구하는 기능 게이트".
- Docker Enterprise와 함께 Kubernetes를 사용하는 경우, "다음 단계에 따라 CLI 액세스를 설정합니다".

다 잡았나요? 좋습니다! 그럼 이제 시작하겠습니다.

Trident 연산자를 구축하고 Hrom을 사용하여 Astra Trident를 설치합니다

나열된 단계를 수행하여 Helm을 사용하여 Trident 연산자를 배포합니다.

필요한 것

위에 나열된 필수 구성 요소 외에도 Hrom을 사용하여 Trident 연산자를 구축하려면 다음이 필요합니다.

- A "지원되는 Kubernetes 버전"
- Helm 버전 3

단계

1. Trident의 제어 리포지토리 추가:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. 를 사용합니다 `helm install` 명령을 입력하고 배포 이름을 지정합니다. 다음 예를 참조하십시오.

```
helm install <name> netapp-trident/trident-operator --version 22.10.0
--create-namespace --namespace <trident-namespace>
```

> Trident에 대한 네임스페이스를 이미 만든 경우 를 참조하십시오 `--create-namespace` 매개 변수는 추가 네임스페이스를 만들지 않습니다.

설치 중에 구성 데이터를 전달하는 방법에는 두 가지가 있습니다.

- `--values` (또는 `-f`): 재정의가 있는 YAML 파일을 지정합니다. 이 옵션은 여러 번 지정할 수 있으며 가장 오른쪽 파일이 우선 적용됩니다.

- `--set`: 명령줄에 재정의를 지정합니다.

예를 들어, 의 기본값을 변경합니다 `debug`에서 다음을 실행합니다 `--set` 명령:

```
helm install <name> netapp-trident/trident-operator --version 22.10.0
--create-namespace --namespace --set tridentDebug=true
```

를 클릭합니다 `values.yaml` 제어 차트의 일부인 파일 은 키 목록과 해당 기본값을 제공합니다.

`helm list` 이름, 네임스페이스, 차트, 상태, 앱 버전, 개정 번호 등

Trident 연산자를 수동으로 구축하고 Trident를 설치합니다

나열된 단계를 수행하여 Trident 연산자를 수동으로 배포합니다.

1단계**: Kubernetes** 클러스터 검증

가장 먼저 해야 할 일은 Linux 호스트에 로그인하여 해당 호스트가 관리 중인 _ 작업 _ 을(를) 관리하고 있는지 확인하는 것입니다. "지원되는 Kubernetes 클러스터" 에 필요한 권한이 있어야 합니다.

OpenShift에서는 을 사용합니다 `oc` 대신 `kubectl` 다음 모든 예에서 를 실행하여 먼저 * system:admin * 으로 로그인합니다 `oc login -u system:admin` 또는 `oc login -u kube-admin`.

Kubernetes 버전을 확인하려면 다음 명령을 실행합니다.

```
kubectl version
```

Kubernetes 클러스터 관리자 권한이 있는지 확인하려면 다음 명령을 실행합니다.

```
kubectl auth can-i '*' '*' --all-namespaces
```

Docker Hub에서 이미지를 사용하는 Pod를 시작하고 Pod 네트워크를 통해 스토리지 시스템에 연결할 수 있는지 확인하려면 다음 명령을 실행합니다.

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

2단계**:** 운영자를 다운로드하고 설정합니다

21.01부터 Trident 연산자는 클러스터 범위입니다. Trident 연산자를 사용하여 Trident를 설치하려면 을 만들어야 합니다 `TridentOrchestrator` 사용자 정의 리소스 정의(CRD) 및 기타 리소스 정의 Astra Trident를 설치하기 전에 다음 단계를 수행하여 운영자를 설정해야 합니다.

1. 에서 최신 버전의 Trident 설치 프로그램 번들을 다운로드하여 압축을 풉니다 "GitHub의 _Assets_섹션".

```
wget
https://github.com/NetApp/trident/releases/download/v22.10.0/trident-
installer-22.10.0.tar.gz
tar -xf trident-installer-22.10.0.tar.gz
cd trident-installer
```

2. 적절한 CRD 매니페스트를 사용하여 를 만듭니다 `TridentOrchestrator` CRD 그런 다음 을 만듭니다 `TridentOrchestrator` 나중에 사용자 지정 리소스를 사용하여 운영자가 설치를 인스턴스화합니다.

   다음 명령을 실행합니다.

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. 를 누릅니다 `TridentOrchestrator` CRD가 생성되면 운용자 배치에 필요한 다음과 같은 리소스를 생성한다.

   - 연산자를 위한 ServiceAccount입니다
   - ServiceAccount에 대한 ClusterRole 및 ClusterRoleBinding
   - 전용 PodSecurityPolicy
   - 작업자 자체

   Trident 설치 프로그램에는 이러한 리소스를 정의하는 매니페스트가 포함되어 있습니다. 기본적으로 연산자는 에 배포됩니다 `trident` 네임스페이스. 를 누릅니다 `trident` 네임스페이스가 없습니다. 다음 매니페스트를 사용하여 네임스페이스를 만듭니다.

```
kubectl apply -f deploy/namespace.yaml
```

4. 기본 네임스페이스 이외의 네임스페이스에 연산자를 배포합니다 `trident` 네임스페이스에서 을 업데이트해야 합니다 `serviceaccount.yaml`, `clusterrolebinding.yaml` 및 `operator.yaml` 을(를) 확인하고 생성합니다 `bundle.yaml`.

   다음 명령을 실행하여 YAML 매니페스트를 업데이트하고 을 생성합니다 `bundle.yaml` 를 사용합니다 `kustomization.yaml`:

```
kubectl kustomize deploy/ > deploy/bundle.yaml
```

   다음 명령을 실행하여 리소스를 생성하고 연산자를 배포합니다.

```
kubectl create -f deploy/bundle.yaml
```

5. 배치한 후 작업자의 상태를 확인하려면 다음을 수행합니다.

```
kubectl get deployment -n <operator-namespace>

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m
```

```
kubectl get pods -n <operator-namespace>

NAME                                READY   STATUS             RESTARTS
AGE
trident-operator-54cb664d-lnjxh     1/1     Running            0
3m
```

운영자 배포는 클러스터의 작업자 노드 중 하나에서 실행되고 있는 포드를 성공적으로 생성합니다.

> Kubernetes 클러스터에는 운영자의 인스턴스 * 하나가 있어야 합니다. Trident 연산자의 여러 배포를 생성하지 마십시오.

3단계**:** 작성 `TridentOrchestrator` **Trident**를 설치합니다

이제 연산자를 사용하여 Astra Trident를 설치할 준비가 되었습니다! 이 작업을 수행하려면 을 만들어야 합니다 `TridentOrchestrator`. Trident 설치 프로그램에는 작성을 위한 예제 정의가 포함되어 있습니다 `TridentOrchestrator`. 그러면 에서 설치가 시작됩니다 `trident` 네임스페이스.

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident
Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:22.10
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:21.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v21.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Trident 연산자를 사용하면 의 특성을 사용하여 Astra Trident가 설치되는 방식을 사용자 지정할 수 있습니다 `TridentOrchestrator` 사양 을 참조하십시오 "Trident 구축을 사용자 지정합니다".

의 상태 `TridentOrchestrator` 설치가 성공적으로 완료되었는지 여부를 나타내고 설치된 Trident의 버전을 표시합니다.

상태	설명
설치 중	이 옵션을 사용하여 Astra Trident를 설치합니다 `TridentOrchestrator` 있습니다.
설치되어 있습니다	Astra Trident가 성공적으로 설치되었습니다.
제거 중	그 이유는 운영자가 Astra Trident를 제거하는 중입니다 `spec.uninstall=true`.
제거되었습니다	Astra Trident가 제거되었습니다.
실패했습니다	운영자가 Astra Trident를 설치, 패치, 업데이트 또는 제거할 수 없습니다. 이 상태에서 자동으로 복구를 시도합니다. 이 상태가 지속되면 문제 해결이 필요합니다.
업데이트 중	운영자가 기존 설치를 업데이트하고 있습니다.
오류	를 클릭합니다 `TridentOrchestrator` 사용되지 않습니다. 다른 파일이 이미 있습니다.

설치하는 동안 의 상태입니다 `TridentOrchestrator` 변경 시작 `Installing` 를 선택합니다 `Installed`. 을(를) 관찰하면 `Failed` 상태 및 운용자가 자체적으로 복구할 수 없는 경우 운용자의 로그를 확인해야 한다. 를 참조하십시오 "문제 해결" 섹션을 참조하십시오.

생성된 포드를 살펴보고 Astra Trident 설치가 완료되었는지 확인할 수 있습니다.

```
kubectl get pod -n trident

NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-7d466bf5c7-v4cpw        5/5     Running   0          1m
trident-csi-mr6zc                   2/2     Running   0          1m
trident-csi-xrp7w                   2/2     Running   0          1m
trident-csi-zh2jt                   2/2     Running   0          1m
trident-operator-766f7b8658-ldzsv   1/1     Running   0          3m
```

을 사용할 수도 있습니다 `tridentctl` 설치된 Astra Trident의 버전을 확인합니다.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.04.0        | 21.04.0        |
+----------------+----------------+
```

이제 백엔드를 생성할 수 있습니다. 을 참조하십시오 "구축 후 작업".

배포 중 문제 해결에 대한 자세한 내용은 을 참조하십시오 "문제 해결" 섹션을 참조하십시오.

Trident 운영자 배포를 사용자 지정합니다

Trident 운영자는 의 특성을 사용하여 Astra Trident 설치를 사용자 지정할 수 있습니다 `TridentOrchestrator` 사양

설치를 사용자 지정하려면 다음을 선택합니다 `TridentOrchestrator` 인수를 사용하면 사용을 고려해야 합니다 `tridentctl` 필요에 따라 수정할 수 있는 사용자 지정 YAML 매니페스트를 생성합니다.

> `spec.namespace` 에 지정됩니다 `TridentOrchestrator` Astra Trident가 설치된 네임스페이스를 나타냅니다. Astra Trident가 설치된 후에는 이 매개 변수 * 를 업데이트할 수 없습니다. 이렇게 하려고 하면 가 발생합니다 `TridentOrchestrator` 변경할 상태입니다 `Failed`. Astra Trident는 네임스페이스 간에 마이그레이션되지 않습니다.

구성 옵션

이 표에 자세히 나와 있습니다 `TridentOrchestrator` 특성:

매개 변수	설명	기본값
`namespace`	Astra Trident를 설치할 네임스페이스입니다	"기본값"
`debug`	Astra Trident에 대한 디버깅을 활성화합니다	거짓
`windows`	를 로 설정합니다 `true` Windows 작업자 노드에 설치할 수 있습니다.	거짓
`IPv6`	IPv6를 통해 Astra Trident를 설치합니다	거짓
`k8sTimeout`	Kubernetes 작업 시간이 초과되었습니다	30초
`silenceAutosupport`	AutoSupport 번들을 NetApp에 자동으로 보내지 않습니다	거짓
`enableNodePrep`	작업자 노드 종속성 자동 관리(* beta*)	거짓
`autosupportImage`	AutoSupport 텔레메트리 컨테이너 이미지입니다	"NetApp/트리덴트 - AutoSupport: 22.10.0"
`autosupportProxy`	AutoSupport 텔레메트리 전송을 위한 프록시의 주소/포트입니다	"http://proxy.example.com:8888"
`uninstall`	Astra Trident를 제거하는 데 사용되는 플래그입니다	거짓
`logFormat`	사용할 Astra Trident 로깅 형식[text,json]	"텍스트"
`tridentImage`	설치할 Astra Trident 이미지	"NetApp/트리덴트: 21.04"

매개 변수	설명	기본값
`imageRegistry`	형식의 내부 레지스트리 경로입니다 `<registry FQDN>[:port][/subpath]`	"k8s.gcr.io/sig-storage(k8s 1.19 이상) 또는 quay.io/k8scsi"
`kubeletDir`	호스트의 kubelet 디렉토리에 대한 경로입니다	“/var/lib/kubelet”
`wipeout`	Astra Trident를 완전히 제거하기 위해 삭제할 리소스 목록입니다	
`imagePullSecrets`	내부 레지스트리에서 이미지를 가져올 수 있는 비밀	
`controllerPluginNodeSelector`	Trident 컨트롤러 CSI 플러그인을 실행하는 Pod용 추가 노드 선택기. pod.spec.nodeSelector 과 동일한 형식을 따릅니다.	기본값 없음, 선택 사항
`controllerPluginTolerations`	Trident 컨트롤러 CSI 플러그인을 실행하는 Pod의 허용 설정을 재정의합니다. pod.spec.Tolerations와 같은 형식을 따릅니다.	기본값 없음, 선택 사항
`nodePluginNodeSelector`	Trident Node CSI 플러그인을 실행하는 Pod용 추가 노드 선택기 pod.spec.nodeSelector 과 동일한 형식을 따릅니다.	기본값 없음, 선택 사항
`nodePluginTolerations`	Trident Node CSI 플러그인을 실행하는 Pod의 허용 설정을 재정의합니다. pod.spec.Tolerations와 같은 형식을 따릅니다.	기본값 없음, 선택 사항

포드 매개 변수 포맷에 대한 자세한 내용은 을 참조하십시오 "노드에 Pod 할당".

샘플 구성

정의할 때 위에서 언급한 속성을 사용할 수 있습니다 `TridentOrchestrator` 를 눌러 설치를 사용자 정의합니다.

예 **1:** 기본 사용자 정의 구성

다음은 기본 사용자 지정 구성의 예입니다.

```
cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
```

예 **2:** 노드 선택기를 사용하여 배포

이 예제에서는 노드 선택기를 사용하여 Trident를 배포하는 방법을 보여 줍니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  controllerPluginNodeSelector:
    nodetype: master
  nodePluginNodeSelector:
    storage: netapp
```

예 **3: Windows** 작업자 노드에 배포

이 예제에서는 Windows 작업자 노드에 대한 배포를 보여 줍니다.

```
$ cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  windows: true
```

tridentctl을 사용하여 배포합니다

을 사용하여 Astra Trident를 배포할 수 있습니다 `tridentctl`. 을 숙지하는 것이 좋습니다 "기본 개념". 를 사용자 지정합니다 `tridentctl` 구축 방법은 을 참조하십시오 "tridentctl 배포를 사용자 지정합니다".

Astra Trident 22.10에 대한 중요 정보입니다

- Astra Trident 22.10으로 업그레이드하기 전에 다음 중요 정보를 읽어야 합니다. *

strong>Astra Trident 22.10에 대한 중요 정보

- 이제 Trident에서 Kubernetes 1.25가 지원됩니다. Kubernetes 1.25로 업그레이드하기 전에 Astra Trident 22.10으로 업그레이드해야 합니다.
- Astra Trident는 이제 SAN 환경에서 다중 경로 구성을 사용하도록 엄격히 적용되며 권장값은 입니다 `find_multipaths: no` 다중 경로 .conf 파일

  비 경로 다중화 구성 또는 의 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트 오류가 발생합니다. Trident에서 의 사용을 권장했습니다 `find_multipaths: no` 21.07 릴리스 이후.

필수 구성 요소를 확인합니다

Astra Trident를 구축하려면 다음과 같은 사전 요구 사항을 충족해야 합니다.

- 지원되는 Kubernetes 클러스터에 대한 모든 권한
- 지원되는 NetApp 스토리지 시스템에 대한 액세스
- 모든 Kubernetes 작업자 노드에서 볼륨을 마운트할 수 있습니다.
- 가 설치된 Linux 호스트 `kubectl` (또는 `oc`, OpenShift를 사용하는 경우) 사용하려는 Kubernetes 클러스터를 관리하도록 설치 및 구성한 것입니다.

- 를 클릭합니다 `KUBECONFIG` 환경 변수는 Kubernetes 클러스터 구성을 가리킵니다.
- 를 클릭합니다 "Astra Trident에서 요구하는 기능 게이트" 가 활성화됩니다.
- Docker Enterprise와 함께 Kubernetes를 사용하는 경우, "다음 단계에 따라 CLI 액세스를 설정합니다".

1단계: Kubernetes 클러스터 검증

Linux 호스트에 로그인하여 작업을 관리하고 있는지 확인합니다. "지원되는 Kubernetes 클러스터" 그리고 필요한 권한이 있습니다.

> OpenShift를 사용하면 을 사용할 수 있습니다 `oc` 대신 `kubectl` 다음 모든 예에서 를 실행하여 먼저 * system:admin * 으로 로그인해야 합니다 `oc login -u system:admin` 또는 `oc login -u kube-admin`.

Kubernetes 버전을 확인하려면 다음 명령을 실행합니다.

```
kubectl version
```

Kubernetes 클러스터 관리자 권한을 확인하려면 다음 명령을 실행합니다.

```
kubectl auth can-i '*' '*' --all-namespaces
```

Docker Hub에서 이미지를 사용하는 Pod를 시작하고 Pod 네트워크를 통해 스토리지 시스템에 연결할 수 있는지 확인하려면 다음 명령을 실행합니다.

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Kubernetes 서버 버전을 식별하십시오. Astra Trident를 설치할 때 사용합니다.

2단계: 설치 프로그램을 다운로드하고 압축을 풉니다

> Trident 설치 관리자는 Trident Pod를 생성하고 상태를 유지하는 데 사용되는 CRD 객체를 구성하며, 클러스터 호스트에 볼륨 프로비저닝 및 연결과 같은 작업을 수행하는 CSI 사이드카를 초기화합니다.

에서 최신 버전의 Trident 설치 프로그램 번들을 다운로드하여 압축을 풀 수 있습니다 "GitHub의 _Assets_섹션".

예를 들어, 최신 버전이 22.10.0인 경우:

```
wget https://github.com/NetApp/trident/releases/download/v22.10.0/trident-
installer-22.10.0.tar.gz
tar -xf trident-installer-22.10.0.tar.gz
cd trident-installer
```

3단계: Astra Trident 설치

를 실행하여 원하는 네임스페이스에 Astra Trident를 설치합니다 `tridentctl install` 명령.

```
./tridentctl install -n trident
....
INFO Starting Trident installation.                namespace=trident
INFO Created service account.
INFO Created cluster role.
INFO Created cluster role binding.
INFO Added finalizers to custom resource definitions.
INFO Created Trident service.
INFO Created Trident secret.
INFO Created Trident deployment.
INFO Created Trident daemonset.
INFO Waiting for Trident pod to start.
INFO Trident pod started.                          namespace=trident
pod=trident-csi-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=22.10.0
INFO Trident installation succeeded.
....
```

Astra Trident를 Windows 노드에서 실행하도록 설정하려면 을 추가합니다 `--windows` 설치 명령에 플래그 지정: `$ ./tridentctl install --windows -n trident`.

설치 프로그램이 완료되면 다음과 유사한 출력이 표시됩니다. Kubernetes 클러스터의 노드 수에 따라 다음과 같은 Pod가 더 많이 있을 수 있습니다.

```
kubectl get pod -n trident
NAME                              READY   STATUS    RESTARTS   AGE
trident-csi-679648bd45-cv2mx   4/4     Running   0          5m29s
trident-csi-vgc8n              2/2     Running   0          5m29s

./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.10.0        | 22.10.0        |
+----------------+----------------+
```

Astra Trident 구성을 완료하려면 로 이동합니다 "구축 후 작업".

설치 프로그램이 성공적으로 완료되지 않거나 `trident-csi-<generated id>` 에는 * Running * 상태가 없으며 플랫폼이 설치되지 않았습니다.

배포 중 문제 해결에 대한 자세한 내용은 을 참조하십시오 "문제 해결".

tridentctl 배포를 사용자 지정합니다

Astra Trident 설치 프로그램을 사용하여 배포를 사용자 지정할 수 있습니다.

설치 프로그램에 대해 알아보십시오

Astra Trident 설치 프로그램을 사용하여 특성을 사용자 지정할 수 있습니다. 예를 들어, Trident 이미지를 개인 저장소에 복사한 경우 를 사용하여 이미지 이름을 지정할 수 있습니다 `--trident-image`. Trident 이미지와 필요한 CSI 사이드카 이미지를 개인 저장소에 복사한 경우 를 사용하여 해당 저장소의 위치를 지정하는 것이 좋습니다 `--image-registry` 스위치를 누릅니다 `<registry FQDN>[:port]`.

Kubernetes 배포를 사용 중인 경우 `kubelet` 일반적인 경로 이외의 경로에 데이터를 보관합니다 `/var/lib/kubelet`, 을 사용하여 대체 경로를 지정할 수 있습니다 `--kubelet-dir`.

설치 관리자의 인수 이외에 설치를 사용자 지정해야 하는 경우 배포 파일을 사용자 지정할 수도 있습니다. 를 사용합니다 `--generate-custom-yaml` 매개 변수는 설치 관리자의 에 다음 YAML 파일을 생성합니다 `setup` 디렉터리:

- `trident-clusterrolebinding.yaml`
- `trident-deployment.yaml`
- `trident-crds.yaml`
- `trident-clusterrole.yaml`
- `trident-daemonset.yaml`
- `trident-service.yaml`
- `trident-namespace.yaml`
- `trident-serviceaccount.yaml`
- `trident-resourcequota.yaml`

이러한 파일을 생성한 후 필요에 따라 수정한 다음 을 사용할 수 있습니다 `--use-custom-yaml` 사용자 지정 배포를 설치합니다.

```
./tridentctl install -n trident --use-custom-yaml
```

다음 단계

Astra Trident를 구축한 후 백엔드 생성, 스토리지 클래스 생성, 볼륨 프로비저닝 및 POD에 볼륨 마운팅으로 진행할 수 있습니다.

1단계: 백엔드를 생성합니다

이제 Astra Trident에서 볼륨을 프로비저닝하는 데 사용할 백엔드를 만들 수 있습니다. 이렇게 하려면 을 만듭니다 `backend.json` 필요한 매개 변수가 포함된 파일입니다. 다양한 백엔드 유형에 대한 샘플 구성 파일은 에서 찾을 수

있습니다 `sample-input` 디렉토리.

을 참조하십시오 "여기" 백엔드 유형에 맞게 파일을 구성하는 방법에 대한 자세한 내용은 을 참조하십시오.

```
cp sample-input/<backend template>.json backend.json
vi backend.json
```

```
./tridentctl -n trident create backend -f backend.json
+-------------+----------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+-------------+----------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+----------------+--------------------------------------
+--------+---------+
```

생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하여 로그를 보고 원인을 확인할 수 있습니다.

```
./tridentctl -n trident logs
```

문제를 해결한 후 이 단계의 처음으로 돌아가서 다시 시도하십시오. 자세한 문제 해결 팁은 를 참조하십시오 "문제 해결" 섹션을 참조하십시오.

2단계: 스토리지 클래스를 생성합니다

Kubernetes 사용자는 가 지정된 영구 PVC(Volume Claim)를 사용하여 볼륨을 프로비저닝합니다 "스토리지 클래스" 이름별. 세부 정보는 사용자로부터 숨겨지지만 스토리지 클래스는 해당 클래스에 사용되는 공급자(이 경우 Trident)와 해당 클래스가 프로비저닝자로부터 의미하는 바를 식별합니다.

Kubernetes 스토리지 클래스를 생성할 때 볼륨을 지정할 시기를 지정하십시오. 수업 구성에서는 이전 단계에서 생성한 백엔드를 모델링해야 하므로 Astra Trident가 이를 사용하여 새 볼륨을 프로비저닝합니다.

가장 간단한 스토리지 클래스는 을 기반으로 합니다 `sample-input/storage-class-csi.yaml.templ` 설치 프로그램과 함께 제공되는 파일로 대치합니다 *`BACKEND_TYPE`* 스토리지 드라이버 이름을 사용합니다.

```
./tridentctl -n trident get backend
+-------------+----------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+----------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+----------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

Kubernetes 오브젝트이므로 를 사용할 수 있습니다 `kubectl` Kubernetes에서 생성해야 합니다.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

이제 Kubernetes 및 Astra Trident에 * basic-CSI * 스토리지 클래스가 표시됩니다. Astra Trident는 백엔드에서 풀을 검색했습니다.

```
kubectl get sc basic-csi
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

3단계: 첫 번째 볼륨을 프로비저닝합니다

이제 첫 번째 볼륨을 동적으로 프로비저닝할 준비가 되었습니다. 이 작업은 Kubernetes를 생성하여 수행합니다 "영구적 볼륨 클레임" (PVC) 개체.

방금 만든 저장소 클래스를 사용하는 볼륨에 대해 PVC를 생성합니다.

을 참조하십시오 `sample-input/pvc-basic-csi.yaml` 예를 들어, 스토리지 클래스 이름이 생성한 이름과 일치하는지 확인합니다.

```
kubectl create -f sample-input/pvc-basic-csi.yaml

kubectl get pvc --watch
NAME      STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS   AGE
basic     Pending
basic          1s
basic     Pending   pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   0
basic          5s
basic     Bound     pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   1Gi
RWO           basic          7s
```

4단계: POD에 볼륨을 마운트합니다

이제 볼륨을 마운트하겠습니다. PV를 장착하는 nginx 포드를 시작합니다 `/usr/share/nginx/html`.

```
cat << EOF > task-pv-pod.yaml
kind: Pod
apiVersion: v1
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
EOF
kubectl create -f task-pv-pod.yaml
```

```
# Wait for the pod to start
kubectl get pod --watch

# Verify that the volume is mounted on /usr/share/nginx/html
kubectl exec -it task-pv-pod -- df -h /usr/share/nginx/html

# Delete the pod
kubectl delete pod task-pv-pod
```

이때 POD(애플리케이션)는 더 이상 존재하지 않지만 볼륨은 여전히 존재합니다. 원하는 경우 다른 포드에서 사용할 수 있습니다.

볼륨을 삭제하려면 클레임을 삭제합니다.

```
kubectl delete pvc basic
```

이제 다음과 같은 추가 작업을 수행할 수 있습니다.

- "추가 백엔드를 구성합니다."
- "추가 스토리지 클래스를 생성합니다."