시작하십시오
Astra Trident

NetApp
November 14, 2025

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

Astra Trident를 설치하기 전에 이러한 일반 시스템 요구 사항을 검토해야 합니다. 특정 백엔드에 추가 요구 사항이 있을 수 있습니다.

Astra Trident 23.01에 대한 중요 정보입니다

- Astra Trident * 에 대한 다음 중요 정보를 읽어야 합니다

**** 중요 정보 **Astra Trident **

- 이제 Trident에서 Kubernetes 1.26이 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 업그레이드하십시오.
- Astra Trident는 SAN 환경에서 다중 경로 구성을 엄격하게 사용하며 권장 값은 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 1.21-1.26

- Mirantis Kubernetes 엔진 3.5
- OpenShift 4.9-4.12

Trident 연산자는 다음 릴리즈에서 지원됩니다.

- Anthos On-Premise(VMware) 및 Anthos의 Bare Metal 1.9, 1.10, 1.11
- Kubernetes 1.21-1.26
- OpenShift 4.9-4.12

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.

Astra Trident가 설치된 Kubernetes 클러스터를 1.24에서 1.25 이상으로 업그레이드하기 전에 을 참조하십시오 "제어 기반 작업자 설치를 업그레이드합니다".

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/9.5 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	1.21-1.26	아니요
볼륨 스냅샷	1.21-1.26	아니요
체적 스냅샷의 PVC	1.21-1.26	아니요
iSCSI PV 크기 조정	1.21-1.26	아니요
ONTAP 양방향 CHAP	1.21-1.26	아니요

피처	**Kubernetes** 버전	기능 게이트가 필요합니까**?**
동적 내보내기 정책	1.21-1.26	아니요
Trident 연산자	1.21-1.26	아니요
CSI 토폴로지	1.21-1.26	아니요

호스트 운영 체제를 테스트했습니다

Astra Trident가 공식적으로 특정 운영 체제를 지원하지 않지만 다음과 같은 운영 체제가 작동하는 것으로 알려져 있습니다.

- OpenShift Container Platform에서 지원하는 RedHat CoreOS(RHCOS) 버전
- RHEL 8+
- Ubuntu 22.04 이상
- 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

Kubernetes 클러스터의 모든 작업자 노드는 Pod용으로 프로비저닝된 볼륨을 마운트할 수 있어야 합니다. 작업자 노드를 준비하려면 드라이버 선택에 따라 NFS 또는 iSCSI 도구를 설치해야 합니다.

"작업자 노드를 준비합니다"

스토리지 시스템 구성

Astra Trident는 백엔드 구성에서 사용하기 전에 스토리지 시스템을 변경해야 할 수 있습니다.

"백엔드 구성"

Astra Trident 포트

Astra Trident는 통신을 위해 특정 포트에 액세스해야 합니다.

"Astra Trident 포트"

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
v1.21.0	• Docker.IO/NetApp/트리덴트: 23.01.1 • Docker.IO/NetApp/트리덴트 - 자동 지원: 23.01 • registry.k8s.io/SIG-storage/CSI-v3.4.0 • registry.k8s.io/SIG-storage/CSI-attacher:v4.1.0 • registry.k8s.io/SIG-storage/CSI-resizer: v1.7.0 • registry.k8s.io/SIG-storage/CSI-snapshotter:v6.2.1 • registry.k8s.io/SIG-storage/CSI-node-driver-registrar: v2.7.0 • Docker.IO/NetApp/트리덴트 - 운영자: 23.01.1(선택 사항)
v1.22.0	• Docker.IO/NetApp/트리덴트: 23.01.1 • Docker.IO/NetApp/트리덴트 - 자동 지원: 23.01 • registry.k8s.io/SIG-storage/CSI-v3.4.0 • registry.k8s.io/SIG-storage/CSI-attacher:v4.1.0 • registry.k8s.io/SIG-storage/CSI-resizer: v1.7.0 • registry.k8s.io/SIG-storage/CSI-snapshotter:v6.2.1 • registry.k8s.io/SIG-storage/CSI-node-driver-registrar: v2.7.0 • Docker.IO/NetApp/트리덴트 - 운영자: 23.01.1(선택 사항)
v1.23.0	• Docker.IO/NetApp/트리덴트: 23.01.1 • Docker.IO/NetApp/트리덴트 - 자동 지원: 23.01 • registry.k8s.io/SIG-storage/CSI-v3.4.0 • registry.k8s.io/SIG-storage/CSI-attacher:v4.1.0 • registry.k8s.io/SIG-storage/CSI-resizer: v1.7.0 • registry.k8s.io/SIG-storage/CSI-snapshotter:v6.2.1 • registry.k8s.io/SIG-storage/CSI-node-driver-registrar: v2.7.0 • Docker.IO/NetApp/트리덴트 - 운영자: 23.01.1(선택 사항)

Kubernetes 버전	컨테이너 이미지
v1.24.0	• Docker.IO/NetApp/트리덴트: 23.01.1 • Docker.IO/NetApp/트리덴트 - 자동 지원: 23.01 • registry.k8s.io/SIG-storage/CSI-v3.4.0 • registry.k8s.io/SIG-storage/CSI-attacher:v4.1.0 • registry.k8s.io/SIG-storage/CSI-resizer: v1.7.0 • registry.k8s.io/SIG-storage/CSI-snapshotter:v6.2.1 • registry.k8s.io/SIG-storage/CSI-node-driver-registrar: v2.7.0 • Docker.IO/NetApp/트리덴트 - 운영자: 23.01.1(선택 사항)
v1.25.0	• Docker.IO/NetApp/트리덴트: 23.01.1 • Docker.IO/NetApp/트리덴트 - 자동 지원: 23.01 • registry.k8s.io/SIG-storage/CSI-v3.4.0 • registry.k8s.io/SIG-storage/CSI-attacher:v4.1.0 • registry.k8s.io/SIG-storage/CSI-resizer: v1.7.0 • registry.k8s.io/SIG-storage/CSI-snapshotter:v6.2.1 • registry.k8s.io/SIG-storage/CSI-node-driver-registrar: v2.7.0 • Docker.IO/NetApp/트리덴트 - 운영자: 23.01.1(선택 사항)
v1.26.0	• Docker.IO/NetApp/트리덴트: 23.01.1 • Docker.IO/NetApp/트리덴트 - 자동 지원: 23.01 • registry.k8s.io/SIG-storage/CSI-v3.4.0 • registry.k8s.io/SIG-storage/CSI-attacher:v4.1.0 • registry.k8s.io/SIG-storage/CSI-resizer: v1.7.0 • registry.k8s.io/SIG-storage/CSI-snapshotter:v6.2.1 • registry.k8s.io/SIG-storage/CSI-node-driver-registrar: v2.7.0 • Docker.IO/NetApp/트리덴트 - 운영자: 23.01.1(선택 사항)

Kubernetes 버전 1.21 이상에서는 검증된 을 사용합니다 `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v6.x` ?곜긽?? `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.

Astra Trident를 설치합니다

Astra Trident 설치에 대해 자세히 알아보십시오

Astra Trident를 다양한 환경과 조직에 설치할 수 있도록 NetApp은 다양한 설치 옵션을 제공합니다. Trident 연산자(수동 또는 Helm 사용) 또는 를 사용하여 Astra Trident를 설치할 수 있습니다 `tridentctl`. 이 항목에서는 적합한 설치 프로세스를 선택하는 데 필요한 중요한 정보를 제공합니다.

Astra Trident 23.01에 대한 중요 정보입니다

- Astra Trident * 에 대한 다음 중요 정보를 읽어야 합니다

**** 중요 정보 **Astra Trident **

- 이제 Trident에서 Kubernetes 1.26이 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 업그레이드하십시오.
- Astra Trident는 SAN 환경에서 다중 경로 구성을 엄격하게 사용하며 권장 값은 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.

시작하기 전에

설치 경로에 관계없이 다음 항목이 있어야 합니다.

- 지원되는 버전의 Kubernetes 및 기능 요구 사항을 실행하는 지원되는 Kubernetes 클러스터에 대한 모든 권한이 활성화됩니다. 를 검토합니다 "요구 사항" 를 참조하십시오.
- 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` Kubernetes 클러스터 구성을 가리키도록 설정된 환경 변수입니다.
- Docker Enterprise와 함께 Kubernetes를 사용하는 경우, "다음 단계에 따라 CLI 액세스를 설정합니다".

에 익숙하지 않은 경우 "기본 개념"이제 아주 좋은 시간입니다.

설치 방법을 선택합니다

적합한 설치 방법을 선택합니다. 의 고려 사항도 검토해야 합니다 "방법 간 이동" 결정을 내리기 전에

Trident 연산자 사용

수동으로 배포하든 Hrom을 사용하건 Trident 운영자는 설치를 단순화하고 Astra Trident 리소스를 동적으로 관리할 수 있는 훌륭한 방법입니다. 물론 가능합니다 "Trident 운영자 배포를 사용자 지정합니다" 의 속성을 사용합니다 `TridentOrchestrator` 사용자 지정 리소스(CR).

Trident 연산자를 사용하면 다음과 같은 이점이 있습니다.

** Astra Trident** 객체 생성 ****

Trident 운영자가 Kubernetes 버전에 대해 다음 오브젝트를 자동으로 생성합니다.

- 운영자용 ServiceAccount입니다
- ServiceAccount에 대한 ClusterRole 및 ClusterRoleBinding
- 전용 PodSecurityPolicy(Kubernetes 1.25 이하)
- 작업자 자체

**** 자동 복구 기능 ****

운영자는 Astra Trident 설치를 모니터링하고 구축이 삭제되거나 실수로 수정된 경우와 같은 문제를 해결하기 위한 조치를 적극적으로 수행합니다. A `trident-operator-<generated-id>` 를 연결하는 POD가 생성됩니다 `TridentOrchestrator` Astra Trident가 설치된 CR. 이렇게 하면 클러스터에 Astra Trident 인스턴스가 하나만 있고 설치가 제어되므로 설치가 매우 강력합니다. 설치 변경(예: 배포 또는 노드 반점 삭제)이 수행되면 운영자가 이를 식별하고 개별적으로 수정합니다.

**** 기존 설치에 대한 간편한 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.

****를 처리하는 **** 자동 **Kubernetes** 업그레이드

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.

** NetApp Console** 사용한 **Kubernetes** 클러스터 관리****

NetApp Console 사용하여 Astra Trident 사용하면 최신 버전의 Astra Trident 로 업그레이드하고, 스토리지 클래스를 추가 및 관리하고 작업 환경에 연결하고, Cloud Backup Service 사용하여 영구 볼륨을 백업할 수 있습니다. 콘솔은 Helm을 사용하거나 수동으로 Trident 운영자를 사용하여 Astra Trident 배포할 수 있도록 지원합니다.

사용 `tridentctl`

업그레이드해야 하는 기존 배포가 있거나 배포를 사용자 지정하려는 경우 고려해야 합니다 . Astra Trident를 구축하는 기존 방법입니다.

가능합니다 Trident 리소스에 대한 매니페스트를 생성합니다. 여기에는 구축, 개발/제거, 서비스 계정, Astra Trident가 설치의 일부로 생성한 클러스터 역할 등이 포함됩니다.

> 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. 적절한 installation_method_를 선택합니다.

설치 모드를 선택합니다

조직에서 요구하는 *installation mode*(Standard, Offline 또는 Remote)를 기반으로 배포 프로세스를 결정합니다.

표준 설치

이것은 Astra Trident를 설치하는 가장 쉬운 방법이며 네트워크 제한이 없는 대부분의 환경에서 작동합니다. 표준 설치 모드에서는 기본 레지스트리를 사용하여 필요한 Trident를 저장합니다 (`docker.io`)와 CSI를 참조하십시오 (`registry.k8s.io`) 이미지.

표준 모드를 사용하는 경우 Astra Trident 설치 프로그램이 다음을 수행합니다.

- 인터넷을 통해 컨테이너 이미지를 가져옵니다
- Kubernetes 클러스터의 모든 적격 노드에서 Astra Trident Pod를 가동하는 구축 또는 노드 데모시작을 생성합니다

오프라인 설치

오프라인 설치 모드는 공기 박수나 안전한 위치에 필요할 수 있습니다. 이 시나리오에서는 필요한 Trident 및 CSI 이미지를 저장하기 위해 단일 전용 미러된 레지스트리 또는 두 개의 미러링된 레지스트리를 만들 수 있습니다.

레지스트리 구성에 관계없이 CSI 이미지는 하나의 레지스트리에 있어야 합니다.

원격 설치

다음은 원격 설치 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.

방법 및 모드에 따라 프로세스를 선택합니다

결정을 내린 후 적절한 프로세스를 선택합니다.

방법	설치 모드
Trident 운영자(수동)	"표준 설치" "오프라인 설치"
Trident 운영자(제어)	"표준 설치" "오프라인 설치"
`tridentctl`	"표준 또는 오프라인 설치"

설치 방법 간 이동

설치 방법을 변경할 수 있습니다. 이렇게 하기 전에 다음 사항을 고려하십시오.

- Astra Trident를 설치 및 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.
- 대신 를 제거하고 대신 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.

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 설치합니다

Trident 연산자 수동 배포(표준 모드)

Trident 연산자를 수동으로 구축하여 Astra Trident를 설치할 수 있습니다. 이 프로세스는 Astra Trident에 필요한 컨테이너 이미지가 개인 레지스트리에 저장되어 있지 않은 설치에 적용됩니다. 개인 이미지 레지스트리가 있는 경우 를 사용합니다 "오프라인 배포를 위한 프로세스입니다".

Astra Trident 23.01에 대한 중요 정보입니다

- Astra Trident * 에 대한 다음 중요 정보를 읽어야 합니다

** 중요 정보 Astra Trident **

- 이제 Trident에서 Kubernetes 1.26이 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 업그레이드하십시오.
- Astra Trident는 SAN 환경에서 다중 경로 구성을 엄격하게 사용하며 권장 값은 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.

검토 "설치 개요" 설치 사전 요구 사항을 충족하고 환경에 맞는 올바른 설치 옵션을 선택했는지 확인합니다.

시작하기 전에

설치를 시작하기 전에 Linux 호스트에 로그인하여 작업 관리 여부를 확인합니다. "지원되는 Kubernetes 클러스터" 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`.

1. Kubernetes 버전 확인:

```
kubectl version
```

2. 클러스터 관리자 권한 확인:

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

3. Docker Hub의 이미지를 사용하는 Pod를 시작하고 Pod 네트워크를 통해 스토리지 시스템에 연결할 수 있는지 확인합니다.

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

1단계**: Trident** 설치 프로그램 패키지를 다운로드합니다

Astra Trident 설치 프로그램 패키지에는 Trident 운영자를 구축하고 Astra Trident를 설치하는 데 필요한 모든 것이 들어 있습니다. 에서 최신 버전의 Trident 설치 프로그램을 다운로드하고 압축을 풉니다 "GitHub의 _Assets_섹션".

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

2단계**:** 을 작성합니다 `TridentOrchestrator` **CRD**

를 생성합니다 `TridentOrchestrator` 사용자 정의 리소스 정의(CRD). 을(를) 생성합니다 `TridentOrchestrator` 나중에 사용자 지정 리소스. 에서 적절한 CRD YAML 버전을 사용하십시오 `deploy/crds` 를 작성합니다 `TridentOrchestrator` CRD

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

3단계**: Trident** 연산자를 배포합니다

Astra Trident 설치 관리자는 연산자를 설치하고 관련 개체를 만드는 데 사용할 수 있는 번들 파일을 제공합니다. 번들 파일은 기본 구성을 사용하여 운영자를 구축하고 Astra Trident를 설치하는 간편한 방법입니다.

- 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`.

Trident 설치 관리자는 에 연산자를 배포합니다 `trident` 네임스페이스. 를 누릅니다 `trident` 네임스페이스가 없습니다. 를 사용하십시오 `kubectl apply -f deploy/namespace.yaml` 를 눌러 만듭니다.

단계

1. 리소스를 생성하고 연산자를 배포합니다.

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

> 를 제외한 네임스페이스에 연산자를 배포합니다 `trident` 네임스페이스, 업데이트 `serviceaccount.yaml`, `clusterrolebinding.yaml` 및 `operator.yaml` 을 사용하여 번들 파일을 생성합니다 `kustomization.yaml`:
>
> ```
> kubectl kustomize deploy/ > deploy/<bundle>.yaml
> ```

2. 작업자가 배치되었는지 확인합니다.

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

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

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

4단계**:** 을 작성합니다 `TridentOrchestrator` **Trident**를 설치합니다

이제 를 만들 수 있습니다 `TridentOrchestrator` Astra Trident를 설치합니다. 필요에 따라 할 수 있습니다 "Trident 설치를 사용자 지정합니다" 의 속성을 사용합니다 `TridentOrchestrator` 사양

```
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:23.01
    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:23.01.1
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v23.01.1
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

설치를 확인합니다

설치를 확인하는 방법에는 여러 가지가 있습니다.

사용 `TridentOrchestrator` 상태

의 상태입니다 `TridentOrchestrator` 설치가 성공적으로 완료되었는지 여부를 나타내고 설치된 Trident의 버전을 표시합니다. 설치하는 동안 의 상태입니다 `TridentOrchestrator` 변경 시작 `Installing` 를 선택합니다 `Installed`. 을(를) 관찰하면 `Failed` 상태 및 운영자가 자체적으로 복구할 수 없습니다. "로그를 확인합니다".

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.

POD 생성 상태 사용

생성된 Pod의 상태를 검토하여 Astra Trident 설치가 완료되었는지 확인할 수 있습니다.

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw        6/6     Running   0
1m
trident-node-linux-mr6zc                   2/2     Running   0
1m
trident-node-linux-xrp7w                   2/2     Running   0
1m
trident-node-linux-zh2jt                   2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv          1/1     Running   0
3m
```

사용 `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

다음 단계

이제 가능합니다 "백엔드 및 스토리지 클래스를 생성하고, 볼륨을 프로비저닝하고, POD에 볼륨을 마운트합니다".

Trident 연산자 수동 배포(오프라인 모드)

Trident 연산자를 수동으로 구축하여 Astra Trident를 설치할 수 있습니다. 이 프로세스는 Astra Trident에 필요한 컨테이너 이미지가 개인 레지스트리에 저장된 설치에 적용됩니다. 개인 이미지 레지스트리가 없는 경우 를 사용합니다 "표준 배포 프로세스".

Astra Trident 23.01에 대한 중요 정보입니다

- Astra Trident * 에 대한 다음 중요 정보를 읽어야 합니다

**** 중요 정보 **Astra Trident **

- 이제 Trident에서 Kubernetes 1.26이 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 업그레이드하십시오.
- Astra Trident는 SAN 환경에서 다중 경로 구성을 엄격하게 사용하며 권장 값은 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 연산자를 수동으로 구축하고 **Trident**를 설치합니다

검토 "설치 개요" 설치 사전 요구 사항을 충족하고 환경에 맞는 올바른 설치 옵션을 선택했는지 확인합니다.

시작하기 전에

Linux 호스트에 로그인하여 작업 및 을 관리하고 있는지 확인합니다 "지원되는 Kubernetes 클러스터" 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`.

1. Kubernetes 버전 확인:

```
kubectl version
```

2. 클러스터 관리자 권한 확인:

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

3. Docker Hub의 이미지를 사용하는 Pod를 시작하고 Pod 네트워크를 통해 스토리지 시스템에 연결할 수 있는지 확인합니다.

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

1단계: **Trident** 설치 프로그램 패키지를 다운로드합니다

Astra Trident 설치 프로그램 패키지에는 Trident 운영자를 구축하고 Astra Trident를 설치하는 데 필요한 모든 것이 들어 있습니다. 에서 최신 버전의 Trident 설치 프로그램을 다운로드하고 압축을 풉니다 "GitHub의 _Assets_섹션".

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

2단계: 을 작성합니다 `TridentOrchestrator` **CRD**

를 생성합니다 `TridentOrchestrator` 사용자 정의 리소스 정의(CRD). 을(를) 생성합니다 `TridentOrchestrator` 나중에 사용자 지정 리소스. 에서 적절한 CRD YAML 버전을 사용하십시오 `deploy/crds` 를 작성합니다 `TridentOrchestrator` CRD:

```
kubectl create -f deploy/crds/<VERSION>.yaml
```

3단계: 운영자의 레지스트리 위치를 업데이트합니다

인치 `/deploy/operator.yaml`, 업데이트 `image: docker.io/netapp/trident-operator:23.01.1` 이미지 레지스트리의 위치를 반영합니다. 귀사의 "Trident 및 CSI 이미지" 하나의 레지스트리 또는 다른 레지스트리에 있을 수 있지만 모든 CSI 이미지는 동일한 레지스트리에 있어야 합니다. 예를 들면 다음과 같습니다.

- `image: <your-registry>/trident-operator:23.01.1` 이미지가 모두 하나의 레지스트리에 있는 경우
- `image: <your-registry>/netapp/trident-operator:23.01.1` Trident 이미지가 CSI 이미지와

다른 레지스트리에 있는 경우

4단계**: Trident** 연산자를 배포합니다

Trident 설치 관리자는 에 연산자를 배포합니다 `trident` 네임스페이스. 를 누릅니다 `trident` 네임스페이스가 없습니다. 를 사용하십시오 `kubectl apply -f deploy/namespace.yaml` 를 눌러 만듭니다.

를 제외한 네임스페이스에 연산자를 배포합니다 `trident` 네임스페이스, 업데이트 `serviceaccount.yaml`, `clusterrolebinding.yaml` 및 `operator.yaml` 오퍼레이터 배치 전.

1. 리소스를 생성하고 연산자를 배포합니다.

```
kubectl kustomize deploy/ > deploy/<BUNDLE>.yaml
```

Astra Trident 설치 관리자는 연산자를 설치하고 관련 개체를 만드는 데 사용할 수 있는 번들 파일을 제공합니다. 번들 파일은 기본 구성을 사용하여 운영자를 구축하고 Astra Trident를 설치하는 간편한 방법입니다.

- 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`.

2. 작업자가 배치되었는지 확인합니다.

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

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

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

5단계**:** 에서 이미지 레지스트리 위치를 업데이트합니다 `TridentOrchestrator`

귀사의 "Trident 및 CSI 이미지" 하나의 레지스트리 또는 다른 레지스트리에 있을 수 있지만 모든 CSI 이미지는 동일한 레지스트리에 있어야 합니다. 업데이트 `deploy/crds/tridentorchestrator_cr.yaml` 레지스트리 구성에 따라 추가 위치 사양을 추가하려면 다음을 수행합니다.

하나의 레지스트리에 있는 이미지

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:23.01"
tridentImage: "<your-registry>/trident:23.01.1"
```

다른 레지스트리의 이미지

추가해야 합니다 `sig-storage` 를 누릅니다 `imageRegistry` 다른 레지스트리 위치를 사용합니다.

```
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:23.01"
tridentImage: "<your-registry>/netapp/trident:23.01.1"
```

6단계**:** 을 작성합니다 `TridentOrchestrator` **Trident**를 설치합니다

이제 를 만들 수 있습니다 `TridentOrchestrator` Astra Trident를 설치합니다. 원하는 경우 더 추가할 수 있습니다 "Trident 설치를 사용자 지정합니다" 의 속성을 사용합니다 `TridentOrchestrator` 사양 다음 예에서는 Trident 및 CSI 이미지가 다른 레지스트리에 있는 설치를 보여 줍니다.

```
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:
  Autosupport Image:  <your-registry>/netapp/trident-autosupport:23.01
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:23.01.1
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/netapp/trident-
autosupport:23.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>/sig-storage
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/netapp/trident:23.01.1
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v23.01.1
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

설치를 확인하는 방법에는 여러 가지가 있습니다.

사용 `TridentOrchestrator` 상태

의 상태입니다 `TridentOrchestrator` 설치가 성공적으로 완료되었는지 여부를 나타내고 설치된 Trident의 버전을 표시합니다. 설치하는 동안 의 상태입니다 `TridentOrchestrator` 변경 시작 `Installing` 를 선택합니다 `Installed`. 을(를) 관찰하면 `Failed` 상태 및 운영자가 자체적으로 복구할 수 없습니다. "로그를 확인합니다".

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.

POD 생성 상태 사용

생성된 Pod의 상태를 검토하여 Astra Trident 설치가 완료되었는지 확인할 수 있습니다.

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw        6/6     Running   0
1m
trident-node-linux-mr6zc                   2/2     Running   0
1m
trident-node-linux-xrp7w                   2/2     Running   0
1m
trident-node-linux-zh2jt                   2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv          1/1     Running   0
3m
```

사용 `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

다음 단계

이제 가능합니다 "백엔드 및 스토리지 클래스를 생성하고, 볼륨을 프로비저닝하고, POD에 볼륨을 마운트합니다".

H제어(표준 모드)를 사용하여 **Trident** 연산자 배포

Trident 연산자를 구축하고 Hrom을 사용하여 Astra Trident를 설치할 수 있습니다. 이 프로세스는 Astra Trident에 필요한 컨테이너 이미지가 개인 레지스트리에 저장되어 있지 않은 설치에 적용됩니다. 개인 이미지 레지스트리가 있는 경우 를 사용합니다 "오프라인 배포를 위한 프로세스입니다".

Astra Trident 23.01에 대한 중요 정보입니다

- Astra Trident * 에 대한 다음 중요 정보를 읽어야 합니다

**** 중요 정보 **Astra Trident **

- 이제 Trident에서 Kubernetes 1.26이 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 업그레이드하십시오.
- Astra Trident는 SAN 환경에서 다중 경로 구성을 엄격하게 사용하며 권장 값은 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 연산자를 구축하고 **Hrom**을 사용하여 **Astra Trident**를 설치합니다

Trident 사용 "Helm 차트" Trident 연산자를 구축하고 Trident를 한 번에 설치할 수 있습니다.

검토 "설치 개요" 설치 사전 요구 사항을 충족하고 환경에 맞는 올바른 설치 옵션을 선택했는지 확인합니다.

시작하기 전에

또한 "구축 사전 요구 사항" 필요한 것입니다 "Helm 버전 3".

단계

1. Astra Trident Helm 리포지토리를 추가합니다.

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

2. 사용 `helm install` 다음 예제에서와 같이 배포 이름을 지정합니다 `23.01.1` 는 설치 중인 Astra Trident의 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace <trident-namespace>
```

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

을 사용할 수 있습니다 `helm list` 이름, 네임스페이스, 차트, 상태, 앱 버전과 같은 설치 세부 정보를 검토하려면 수정본 번호.

설치 중에 구성 데이터를 전달합니다

설치 중에 구성 데이터를 전달하는 방법에는 두 가지가 있습니다.

옵션을 선택합니다	설명
`--values` (또는 `-f`)	재정의가 있는 YAML 파일을 지정합니다. 이 옵션은 여러 번 지정할 수 있으며 가장 오른쪽 파일이 우선 적용됩니다.
`--set`	명령줄에 overrides를 지정합니다.

예를 들어, 의 기본값을 변경합니다 `debug`에서 다음을 실행합니다 `--set` 명령 위치 `23.01.1` 설치 중인 Astra Trident의 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --set tridentDebug=true
```

구성 옵션

이 표와 `values.yaml` 제어 차트의 일부인 파일 에는 키 목록과 해당 기본값이 나와 있습니다.

옵션을 선택합니다	설명	기본값
`nodeSelector`	POD 할당을 위한 노드 레이블입니다	
`podAnnotations`	창 주석	
`deploymentAnnotations`	배포 주석	

옵션을 선택합니다	설명	기본값
`tolerations`	POD 지정에 대한 공차	
`affinity`	POD 할당에 대한 선호도	
`tridentControllerPluginNodeSelector`	Pod용 추가 노드 선택기 을 참조하십시오 컨트롤러 Pod 및 노드 포드 이해 를 참조하십시오.	
`tridentControllerPluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 을 참조하십시오 컨트롤러 Pod 및 노드 포드 이해 를 참조하십시오.	
`tridentNodePluginNodeSelector`	Pod용 추가 노드 선택기 을 참조하십시오 컨트롤러 Pod 및 노드 포드 이해 를 참조하십시오.	
`tridentNodePluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 을 참조하십시오 컨트롤러 Pod 및 노드 포드 이해 를 참조하십시오.	
`imageRegistry`	에 대한 레지스트리를 식별합니다 `trident-operator`, `trident`및 기타 이미지. 기본값을 그대로 사용하려면 비워 두십시오.	""
`imagePullPolicy`	에 대한 이미지 풀 정책을 설정합니다 `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	의 이미지 풀 비밀을 설정합니다 `trident-operator`, `trident`및 기타 이미지.	
`kubeletDir`	kubelet 내부 상태의 호스트 위치를 재정의할 수 있습니다.	"/var/lib/kubelet"
`operatorLogLevel`	Trident 연산자의 로그 수준을 다음으로 설정할 수 있습니다. `trace`, `debug`, `info`, `warn`, `error`, 또는 `fatal`.	`"info"`
`operatorDebug`	Trident 연산자의 로그 수준을 디버깅으로 설정할 수 있습니다.	`true`
`operatorImage`	에 대한 이미지를 완전히 재정의할 수 있습니다 `trident-operator`.	""
`operatorImageTag`	의 태그를 재정의할 수 있습니다 `trident-operator` 이미지.	""
`tridentIPv6`	Astra Trident가 IPv6 클러스터에서 작동하도록 허용합니다.	`false`
`tridentK8sTimeout`	대부분의 Kubernetes API 작업에 대한 기본 30초 시간 초과(0이 아닌 경우 초)를 재정의합니다.	`0`

옵션을 선택합니다	설명	기본값
`tridentHttpRequestTimeout`	에서는 HTTP 요청에 대한 기본 90초 제한 시간을 재정의합니다 `0s` 제한 시간 동안 무한 지속 시간입니다. 음수 값은 허용되지 않습니다.	`"90s"`
`tridentSilenceAutosupport`	Astra Trident Periodic AutoSupport 보고를 비활성화할 수 있습니다.	`false`
`tridentAutosupportImageTag`	Astra Trident AutoSupport 컨테이너의 이미지 태그를 재정의할 수 있습니다.	`<version>`
`tridentAutosupportProxy`	Astra Trident AutoSupport 컨테이너가 HTTP 프록시를 통해 집에 전화를 걸 수 있도록 허용합니다.	""
`tridentLogFormat`	Astra Trident 로깅 형식을 설정합니다 (`text` 또는 `json`)를 클릭합니다.	`"text"`
`tridentDisableAuditLog`	Astra Trident 감사 로거를 비활성화합니다.	`true`
`tridentLogLevel`	Astra Trident의 로그 수준을 다음과 같이 설정할 수 있습니다. `trace`, `debug`, `info`, `warn`, `error`, 또는 `fatal`.	`"info"`
`tridentDebug`	Astra Trident의 로그 수준을 로 설정할 수 있습니다 `debug`.	`false`
`tridentLogWorkflows`	추적 로깅 또는 로그 억제를 위해 특정 Astra Trident 워크플로우를 활성화할 수 있습니다.	""
`tridentLogLayers`	추적 로깅 또는 로그 억제를 위해 특정 Astra Trident 계층을 활성화할 수 있습니다.	""
`tridentImage`	Astra Trident의 이미지를 완전히 재정의할 수 있습니다.	""
`tridentImageTag`	Astra Trident에 대한 이미지 태그를 재정의할 수 있습니다.	""
`tridentProbePort`	Kubernetes 활성/준비 프로브에 사용되는 기본 포트를 재정의할 수 있습니다.	""
`windows`	Windows 작업자 노드에 Astra Trident를 설치할 수 있습니다.	`false`
`enableForceDetach`	힘 분리 기능을 활성화합니다.	`false`
`excludePodSecurityPolicy`	운영자 POD 보안 정책을 생성할 수 없습니다.	`false`

컨트롤러 **Pod** 및 노드 포드 이해

Astra Trident는 단일 컨트롤러 POD와 클러스터의 각 작업자 노드에 노드 POD를 더한 형태로 실행됩니다. Astra Trident 볼륨을 마운트하려는 호스트에서 노드 포드가 실행되고 있어야 합니다.

쿠버네티스 "노드 선택기" 및 "관용과 오해" 포드를 특정 노드 또는 기본 노드에서 실행하도록 제한하는 데 사용됩니다. ControllerPlugin과 을 사용합니다 `NodePlugin`구속 조건과 덮어쓰기를 지정할 수 있습니다.

- 컨트롤러 플러그인은 스냅샷 및 크기 조정과 같은 볼륨 프로비저닝 및 관리를 처리합니다.
- 노드 플러그인은 스토리지에 노드를 연결하는 작업을 처리합니다.

다음 단계

이제 가능합니다 "백엔드 및 스토리지 클래스를 생성하고, 볼륨을 프로비저닝하고, POD에 볼륨을 마운트합니다".

H제어(오프라인 모드)를 사용하여 **Trident** 연산자 배포

Trident 연산자를 구축하고 Hrom을 사용하여 Astra Trident를 설치할 수 있습니다. 이 프로세스는 Astra Trident에 필요한 컨테이너 이미지가 개인 레지스트리에 저장된 설치에 적용됩니다. 개인 이미지 레지스트리가 없는 경우 를 사용합니다 "표준 배포 프로세스".

Astra Trident 23.01에 대한 중요 정보입니다

- Astra Trident * 에 대한 다음 중요 정보를 읽어야 합니다

**** 중요 정보 **Astra Trident **

- 이제 Trident에서 Kubernetes 1.26이 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 업그레이드하십시오.
- Astra Trident는 SAN 환경에서 다중 경로 구성을 엄격하게 사용하며 권장 값은 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 연산자를 구축하고 **Hrom**을 사용하여 **Astra Trident**를 설치합니다

Trident 사용 "Helm 차트" Trident 연산자를 구축하고 Trident를 한 번에 설치할 수 있습니다.

검토 "설치 개요" 설치 사전 요구 사항을 충족하고 환경에 맞는 올바른 설치 옵션을 선택했는지 확인합니다.

시작하기 전에

또한 "구축 사전 요구 사항" 필요한 것입니다 "Helm 버전 3".

단계

1. Astra Trident Helm 리포지토리를 추가합니다.

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

2. 사용 `helm install` 배포 및 이미지 레지스트리 위치의 이름을 지정합니다. 귀사의 "Trident 및 CSI 이미지" 하나의 레지스트리 또는 다른 레지스트리에 있을 수 있지만 모든 CSI 이미지는 동일한 레지스트리에 있어야 합니다. 예를 들어, `23.01.1` 는 설치 중인 Astra Trident의 버전입니다.

하나의 레지스트리에 있는 이미지

```
helm install <name> netapp-trident/trident-operator --version
23.01.1 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace>
```

다른 레지스트리의 이미지

추가해야 합니다 `sig-storage` 를 누릅니다 `imageRegistry` 다른 레지스트리 위치를 사용합니다.

```
helm install <name> netapp-trident/trident-operator --version
23.01.1 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.01.1 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.01 --set tridentImage=<your-
registry>/netapp/trident:23.01.1 --create-namespace --namespace
<trident-namespace>
```

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

을 사용할 수 있습니다 `helm list` 이름, 네임스페이스, 차트, 상태, 앱 버전과 같은 설치 세부 정보를 검토하려면 수정본 번호.

설치 중에 구성 데이터를 전달합니다

설치 중에 구성 데이터를 전달하는 방법에는 두 가지가 있습니다.

옵션을 선택합니다	설명
`--values` (또는 `-f`)	재정의가 있는 YAML 파일을 지정합니다. 이 옵션은 여러 번 지정할 수 있으며 가장 오른쪽 파일이 우선 적용됩니다.
`--set`	명령줄에 overrides를 지정합니다.

예를 들어, 의 기본값을 변경합니다 `debug`에서 다음을 실행합니다 `--set` 명령 위치 `23.01.1` 설치 중인 Astra Trident의 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --set tridentDebug=true
```

구성 옵션

이 표와 `values.yaml` 제어 차트의 일부인 파일 에는 키 목록과 해당 기본값이 나와 있습니다.

옵션을 선택합니다	설명	기본값
`nodeSelector`	POD 할당을 위한 노드 레이블입니다	
`podAnnotations`	창 주석	
`deploymentAnnotations`	배포 주석	
`tolerations`	POD 지정에 대한 공차	
`affinity`	POD 할당에 대한 선호도	
`tridentControllerPluginNodeSelector`	Pod용 추가 노드 선택기 을 참조하십시오 컨트롤러 Pod 및 노드 포드 이해 를 참조하십시오.	
`tridentControllerPluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 을 참조하십시오 컨트롤러 Pod 및 노드 포드 이해 를 참조하십시오.	
`tridentNodePluginNodeSelector`	Pod용 추가 노드 선택기 을 참조하십시오 컨트롤러 Pod 및 노드 포드 이해 를 참조하십시오.	
`tridentNodePluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 을 참조하십시오 컨트롤러 Pod 및 노드 포드 이해 를 참조하십시오.	
`imageRegistry`	에 대한 레지스트리를 식별합니다 `trident-operator`, `trident`및 기타 이미지. 기본값을 그대로 사용하려면 비워 두십시오.	""
`imagePullPolicy`	에 대한 이미지 풀 정책을 설정합니다 `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	의 이미지 풀 비밀을 설정합니다 `trident-operator`, `trident`및 기타 이미지.	
`kubeletDir`	kubelet 내부 상태의 호스트 위치를 재정의할 수 있습니다.	"/var/lib/kubelet"
`operatorLogLevel`	Trident 연산자의 로그 수준을 다음으로 설정할 수 있습니다. `trace`, `debug`, `info`, `warn`, `error`, 또는 `fatal`.	`"info"`

옵션을 선택합니다	설명	기본값
`operatorDebug`	Trident 연산자의 로그 수준을 디버깅으로 설정할 수 있습니다.	`true`
`operatorImage`	에 대한 이미지를 완전히 재정의할 수 있습니다 `trident-operator`.	""
`operatorImageTag`	의 태그를 재정의할 수 있습니다 `trident-operator` 이미지.	""
`tridentIPv6`	Astra Trident가 IPv6 클러스터에서 작동하도록 허용합니다.	`false`
`tridentK8sTimeout`	대부분의 Kubernetes API 작업에 대한 기본 30초 시간 초과(0이 아닌 경우 초)를 재정의합니다.	`0`
`tridentHttpRequestTimeout`	에서는 HTTP 요청에 대한 기본 90초 제한 시간을 재정의합니다 `0s` 제한 시간 동안 무한 지속 시간입니다. 음수 값은 허용되지 않습니다.	`"90s"`
`tridentSilenceAutosupport`	Astra Trident Periodic AutoSupport 보고를 비활성화할 수 있습니다.	`false`
`tridentAutosupportImageTag`	Astra Trident AutoSupport 컨테이너의 이미지 태그를 재정의할 수 있습니다.	`<version>`
`tridentAutosupportProxy`	Astra Trident AutoSupport 컨테이너가 HTTP 프록시를 통해 집에 전화를 걸 수 있도록 허용합니다.	""
`tridentLogFormat`	Astra Trident 로깅 형식을 설정합니다 (`text` 또는 `json`)를 클릭합니다.	`"text"`
`tridentDisableAuditLog`	Astra Trident 감사 로거를 비활성화합니다.	`true`
`tridentLogLevel`	Astra Trident의 로그 수준을 다음과 같이 설정할 수 있습니다. `trace`, `debug`, `info`, `warn`, `error`, 또는 `fatal`.	`"info"`
`tridentDebug`	Astra Trident의 로그 수준을 로 설정할 수 있습니다 `debug`.	`false`
`tridentLogWorkflows`	추적 로깅 또는 로그 억제를 위해 특정 Astra Trident 워크플로우를 활성화할 수 있습니다.	""
`tridentLogLayers`	추적 로깅 또는 로그 억제를 위해 특정 Astra Trident 계층을 활성화할 수 있습니다.	""
`tridentImage`	Astra Trident의 이미지를 완전히 재정의할 수 있습니다.	""
`tridentImageTag`	Astra Trident에 대한 이미지 태그를 재정의할 수 있습니다.	""

옵션을 선택합니다	설명	기본값
`tridentProbePort`	Kubernetes 활성/준비 프로브에 사용되는 기본 포트를 재정의할 수 있습니다.	""
`windows`	Windows 작업자 노드에 Astra Trident를 설치할 수 있습니다.	`false`
`enableForceDetach`	힘 분리 기능을 활성화합니다.	`false`
`excludePodSecurityPolicy`	운영자 POD 보안 정책을 생성할 수 없습니다.	`false`

컨트롤러 **Pod** 및 노드 포드 이해

Astra Trident는 단일 컨트롤러 POD와 클러스터의 각 작업자 노드에 노드 POD를 더한 형태로 실행됩니다. Astra Trident 볼륨을 마운트하려는 호스트에서 노드 포드가 실행되고 있어야 합니다.

쿠버네티스 "노드 선택기" 및 "관용과 오해" 포드를 특정 노드 또는 기본 노드에서 실행하도록 제한하는 데 사용됩니다. ControllerPlugin과 을 사용합니다 `NodePlugin`구속 조건과 덮어쓰기를 지정할 수 있습니다.

- 컨트롤러 플러그인은 스냅샷 및 크기 조정과 같은 볼륨 프로비저닝 및 관리를 처리합니다.
- 노드 플러그인은 스토리지에 노드를 연결하는 작업을 처리합니다.

다음 단계

이제 가능합니다 "백엔드 및 스토리지 클래스를 생성하고, 볼륨을 프로비저닝하고, POD에 볼륨을 마운트합니다".

Trident 운영자 설치를 사용자 지정합니다

Trident 운영자는 의 특성을 사용하여 Astra Trident 설치를 사용자 지정할 수 있습니다 `TridentOrchestrator` 사양 설치를 사용자 지정하려면 다음을 선택합니다 `TridentOrchestrator` 인수를 사용할 수 있습니다. 을 사용하는 것이 좋습니다 `tridentctl` 필요에 따라 수정할 사용자 지정 YAML 매니페스트를 생성합니다.

컨트롤러 **Pod** 및 노드 포드 이해

Astra Trident는 단일 컨트롤러 POD와 클러스터의 각 작업자 노드에 노드 POD를 더한 형태로 실행됩니다. Astra Trident 볼륨을 마운트하려는 호스트에서 노드 포드가 실행되고 있어야 합니다.

쿠버네티스 "노드 선택기" 및 "관용과 오해" 포드를 특정 노드 또는 기본 노드에서 실행하도록 제한하는 데 사용됩니다. ControllerPlugin과 을 사용합니다 `NodePlugin`구속 조건과 덮어쓰기를 지정할 수 있습니다.

- 컨트롤러 플러그인은 스냅샷 및 크기 조정과 같은 볼륨 프로비저닝 및 관리를 처리합니다.
- 노드 플러그인은 스토리지에 노드를 연결하는 작업을 처리합니다.

구성 옵션

`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.

이 표에 자세히 나와 있습니다 `TridentOrchestrator` 속성.

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: 23.01"
`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"
`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	

매개 변수	설명	기본값
`imagePullPolicy`	Trident 운영자의 이미지 풀 정책을 설정합니다. 유효한 값은 다음과 같습니다. `Always` 항상 이미지를 당깁니다. `IfNotPresent` 이미지가 아직 노드에 없는 경우에만 이미지를 가져옵니다. `Never` 이미지를 당기지 않습니다.	`IfNotPresent`
`controllerPluginNodeSelector`	Pod용 추가 노드 선택기 pod.spec.nodeSelector 과 동일한 형식을 따릅니다.	기본값 없음, 선택 사항
`controllerPluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. pod.spec.Tolerations와 같은 형식을 따릅니다.	기본값 없음, 선택 사항
`nodePluginNodeSelector`	Pod용 추가 노드 선택기 pod.spec.nodeSelector 과 동일한 형식을 따릅니다.	기본값 없음, 선택 사항
`nodePluginTolerations`	Pod에 대한 Kubernetes 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 설치합니다

tridentctl을 사용하여 설치합니다

을 사용하여 Astra Trident를 설치할 수 있습니다 `tridentctl`. 이 프로세스는 Astra Trident에 필요한 컨테이너 이미지가 개인 레지스트리에 저장되거나 저장되지 않은 설치에 적용됩니다. 를 사용자 지정합니다 `tridentctl` 구축 방법은 을 참조하십시오 "tridentctl 배포를 사용자 지정합니다".

Astra Trident 23.01에 대한 중요 정보입니다

- Astra Trident * 에 대한 다음 중요 정보를 읽어야 합니다

**** 중요 정보 **Astra Trident **

- 이제 Trident에서 Kubernetes 1.26이 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 업그레이드하십시오.
- Astra Trident는 SAN 환경에서 다중 경로 구성을 엄격하게 사용하며 권장 값은 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.

를 사용하여 **Astra Trident**를 설치합니다 `tridentctl`

검토 "설치 개요" 설치 사전 요구 사항을 충족하고 환경에 맞는 올바른 설치 옵션을 선택했는지 확인합니다.

시작하기 전에

설치를 시작하기 전에 Linux 호스트에 로그인하여 작업 관리 여부를 확인합니다. "지원되는 Kubernetes 클러스터" 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`.

1. Kubernetes 버전 확인:

```
kubectl version
```

2. 클러스터 관리자 권한 확인:

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

3. Docker Hub의 이미지를 사용하는 Pod를 시작하고 Pod 네트워크를 통해 스토리지 시스템에 연결할 수 있는지 확인합니다.

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

1단계**: Trident** 설치 프로그램 패키지를 다운로드합니다

Astra Trident 설치 프로그램 패키지는 Trident Pod를 생성하고 상태를 유지하는 데 사용되는 CRD 객체를 구성하며, CSI 사이드카를 초기화하여 클러스터 호스트에 볼륨 프로비저닝 및 연결과 같은 작업을 수행합니다. 에서 최신 버전의 Trident 설치 프로그램을 다운로드하고 압축을 풉니다 "GitHub의 *Assets*_섹션". 선택한 Astra Trident 버전을 사용하여

예제에서 update<trident-installer-XX.XX.X.tar.gz>_를 선택합니다.

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

2단계: **Astra Trident** 설치

를 실행하여 원하는 네임스페이스에 Astra Trident를 설치합니다 `tridentctl install` 명령. 추가 인수를 추가하여 이미지 레지스트리 위치를 지정할 수 있습니다.

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

표준 모드

```
./tridentctl install -n trident
```

하나의 레지스트리에 있는 이미지

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:23.01 --trident
-image <your-registry>/trident:23.01.1
```

다른 레지스트리의 이미지

추가해야 합니다 `sig-storage` 를 누릅니다 `imageRegistry` 다른 레지스트리 위치를 사용합니다.

```
./tridentctl install -n trident --image-registry <your-registry>/sig-
storage --autosupport-image <your-registry>/netapp/trident-
autosupport:23.01 --trident-image <your-
registry>/netapp/trident:23.01.1
```

설치 상태는 다음과 같습니다.

```
....
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-controller-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=23.01.1
INFO Trident installation succeeded.
....
```

설치를 확인합니다

POD 생성 상태 또는 를 사용하여 설치를 확인할 수 있습니다 `tridentctl`.

POD 생성 상태 사용

생성된 Pod의 상태를 검토하여 Astra Trident 설치가 완료되었는지 확인할 수 있습니다.

```
kubectl get pods -n trident

NAME                                  READY   STATUS    RESTARTS   AGE
trident-controller-679648bd45-cv2mx   6/6     Running   0          5m29s
trident-node-linux-vgc8n              2/2     Running   0          5m29s
```

설치 프로그램이 성공적으로 완료되지 않거나 `trident-controller-<generated id>` (`trident-csi-<generated id>` 23.01 이전 버전에서는 * Running * 상태가 없으며 플랫폼이 설치되지 않았습니다. 사용 `-d` 를 선택합니다 "디버그 모드를 켭니다" 문제를 해결합니다.

사용 `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

다음 단계

이제 가능합니다 "백엔드 및 스토리지 클래스를 생성하고, 볼륨을 프로비저닝하고, POD에 볼륨을 마운트합니다".

tridentctl 설치를 사용자 지정합니다

Astra Trident 설치 프로그램을 사용하여 설치를 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 설치한 후 백엔드 생성, 스토리지 클래스 생성, 볼륨 프로비저닝 및 POD에 볼륨 마운팅을 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."

저작권 정보

상표 정보