Trident 연산자를 사용하여 설치합니다

Astra Trident

NetApp
June 28, 2024

목차

Trident 연산자를 사용하여 설치합니다 1
- Trident 연산자 수동 배포(표준 모드) 1
- Trident 연산자 수동 배포(오프라인 모드) 6
- H제어(표준 모드)를 사용하여 Trident 연산자 배포 12
- H제어(오프라인 모드)를 사용하여 Trident 연산자 배포 16
- Trident 운영자 설치를 사용자 지정합니다 20

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 24.02에 대한 중요 정보

- Astra Trident * 에 대한 다음 중요 정보를 읽어야 합니다

** 중요 정보 Astra Trident **

- 이제 Trident에서 Kubernetes 1.27이 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 Trident를 설치합니다

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/v24.02.0/trident-
installer-24.02.0.tar.gz
tar -xf trident-installer-24.02.0.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 버전을 실행하는 클러스터의 경우, 를 사용합니다 `bundle_pre_1_25.yaml`.
- Kubernetes 1.25 이상을 실행하는 클러스터의 경우 를 사용합니다 `bundle_post_1_25.yaml`.

시작하기 전에

- 기본적으로 Trident 설치 관리자는 에 연산자를 배포합니다 `trident` 네임스페이스. 를 누릅니다 `trident` 네임스페이스가 없습니다. 다음을 사용하여 생성합니다.

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

- 를 제외한 네임스페이스에 연산자를 배포합니다 `trident` 네임스페이스, 업데이트 `serviceaccount.yaml`, `clusterrolebinding.yaml` 및 `operator.yaml` 을 사용하여 번들 파일을 생성합니다 `kustomization.yaml`.

  a. 를 생성합니다 `kustomization.yaml` 다음 명령을 사용합니다. 여기서 _<bundle.yaml>_is `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml` Kubernetes 버전을 기반으로 합니다.

```
cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
```

  b. 다음 명령을 사용하여 번들을 컴파일합니다. 여기서 _<bundle.yaml>_is `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml` Kubernetes 버전을 기반으로 합니다.

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

단계

1. 리소스를 생성하고 연산자를 배포합니다.

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

2. 운영자, 배포 및 복제 생성 여부를 확인합니다.

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

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:24.02
    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:24.02.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v24.02.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

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 설치, 패치, 업데이트 또는 제거할 수 없습니다 Astra Trident: 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 것도 이미 있습니다 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 |
+----------------+----------------+
| 24.02.0        | 24.02.0        |
+----------------+----------------+
```

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 24.02에 대한 중요 정보

- Astra Trident * 에 대한 다음 중요 정보를 읽어야 합니다

** 중요 정보 Astra Trident **

- 이제 Trident에서 Kubernetes 1.27이 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 Trident를 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/v24.02.0/trident-
installer-24.02.0.tar.gz
tar -xf trident-installer-24.02.0.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:24.02.0` 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:24.02.0` 이미지가 모두 하나의 레지스트리에 있는 경우
- `image: <your-registry>/netapp/trident-operator:24.02.0` Trident 이미지가 CSI 이미지와

다른 레지스트리에 있는 경우

4단계**: 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 버전을 실행하는 클러스터의 경우, 를 사용합니다 `bundle_pre_1_25.yaml`.
- Kubernetes 1.25 이상을 실행하는 클러스터의 경우 를 사용합니다 `bundle_post_1_25.yaml`.

시작하기 전에

- 기본적으로 Trident 설치 관리자는 에 연산자를 배포합니다 `trident` 네임스페이스. 를 누릅니다 `trident` 네임스페이스가 없습니다. 다음을 사용하여 생성합니다.

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

- 를 제외한 네임스페이스에 연산자를 배포합니다 `trident` 네임스페이스, 업데이트 `serviceaccount.yaml`, `clusterrolebinding.yaml` 및 `operator.yaml` 을 사용하여 번들 파일을 생성합니다 `kustomization.yaml`.

  a. 를 생성합니다 `kustomization.yaml` 다음 명령을 사용합니다. 여기서 _<bundle.yaml>_is `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml` Kubernetes 버전을 기반으로 합니다.

  ```
  cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
  ```

  b. 다음 명령을 사용하여 번들을 컴파일합니다. 여기서 _<bundle.yaml>_is `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml` Kubernetes 버전을 기반으로 합니다.

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

단계

1. 리소스를 생성하고 연산자를 배포합니다.

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

2. 운영자, 배포 및 복제 생성 여부를 확인합니다.

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

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:24.02"
tridentImage: "<your-registry>/trident:24.02.0"
```

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:24.02"
tridentImage: "<your-registry>/netapp/trident:24.02.0"
```

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:24.02
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:24.02.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/netapp/trident-
autosupport:24.02
    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:24.02.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v24.02.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

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 설치, 패치, 업데이트 또는 제거할 수 없습니다 Astra Trident: 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 것도 이미 있습니다 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 |
+----------------+----------------+
| 24.02.0        | 24.02.0        |
+----------------+----------------+
```

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 24.02에 대한 중요 정보

- Astra Trident * 에 대한 다음 중요 정보를 읽어야 합니다

** 중요 정보 Astra Trident **

- 이제 Trident에서 Kubernetes 1.27이 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 Hrom을 사용하여 Astra Trident를 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 `100.2402.0` 는 설치 중인 Astra Trident의 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 100.2402.0
--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 `100.2402.0` 설치 중인 Astra Trident의 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 100.2402.0
--create-namespace --namespace trident --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	

옵션을 선택합니다	설명	기본값
`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`
`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.	`""`

옵션을 선택합니다	설명	기본값
`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`
`cloudProvider`	를 로 설정합니다 `"Azure"` AKS 클러스터에서 관리되는 ID 또는 클라우드 ID를 사용하는 경우 EKS 클러스터에서 클라우드 ID를 사용하는 경우 "AWS"로 설정합니다.	`""`
`cloudIdentity`	AKS 클러스터에서 클라우드 ID를 사용할 때 워크로드 ID("Azure.workload.identity/client-id: xxxxxxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx")로 설정합니다. EKS 클러스터에서 클라우드 ID를 사용할 때 AWS IAM 역할("'eks.amazonaws.com/role-arn: arn:AWS:IAM::123456:role/astratrident-role'")으로 설정합니다.	`""`
`iscsiSelfHealingInterval`	iSCSI 자동 복구가 호출되는 간격입니다.	`5m0s`
`iscsiSelfHealingWaitTime`	iSCSI 자체 복구가 로그아웃과 후속 로그인을 수행하여 부실 세션을 해결하려는 시도를 시작한 이후의 기간입니다.	`7m0s`

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.

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 24.02에 대한 중요 정보

- Astra Trident * 에 대한 다음 중요 정보를 읽어야 합니다

** 중요 정보 Astra Trident **

- 이제 Trident에서 Kubernetes 1.27이 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 Hrom을 사용하여 Astra Trident를 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, `100.2402.0` 는 설치 중인 Astra Trident의 버전입니다.

하나의 레지스트리에 있는 이미지

```
helm install <name> netapp-trident/trident-operator --version
100.2402.0 --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
100.2402.0 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:24.02.0 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:24.02 --set tridentImage=<your-
registry>/netapp/trident:24.02.0 --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 `100.2402.0` 설치 중인 Astra Trident의 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 100.2402.0
--create-namespace --namespace trident --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"`
`operatorDebug`	Trident 연산자의 로그 수준을 디버깅으로 설정할 수 있습니다.	`true`
`operatorImage`	에 대한 이미지를 완전히 재정의할 수 있습니다 `trident-operator`.	""

옵션을 선택합니다	설명	기본값
`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.	""
`tridentProbePort`	Kubernetes 활성/준비 프로브에 사용되는 기본 포트를 재정의할 수 있습니다.	""

옵션을 선택합니다	설명	기본값
`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`

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	`"default"`
`debug`	Astra Trident에 대한 디버깅을 활성화합니다	`false`
`enableForceDetach`	`ontap-san` 및 `ontap-san-economy` 만 해당. Kubernetes Non-Graceful Node Shutdown(ngns)과 함께 작동하여 클러스터 관리자가 마운트된 볼륨이 있는 워크로드를 노드가 정상 상태가 아닌 경우 새 노드로 안전하게 마이그레이션할 수 있도록 합니다.	`false`

매개 변수	설명	기본값
`windows`	를 로 설정합니다 `true` Windows 작업자 노드에 설치할 수 있습니다.	`false`
`cloudProvider`	를 로 설정합니다 "`Azure`" AKS 클러스터에서 관리되는 ID 또는 클라우드 ID를 사용하는 경우 EKS 클러스터에서 클라우드 ID를 사용하는 경우 "AWS"로 설정합니다.	""
`cloudIdentity`	AKS 클러스터에서 클라우드 ID를 사용할 때 워크로드 ID("Azure.workload.identity/client-id: xxxxxxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx")로 설정합니다. EKS 클러스터에서 클라우드 ID를 사용할 때 AWS IAM 역할("'eks.amazonaws.com/role-arn: arn:AWS:IAM::123456:role/astratrident-role")으로 설정합니다.	""
`IPv6`	IPv6를 통해 Astra Trident를 설치합니다	거짓
`k8sTimeout`	Kubernetes 작업 시간이 초과되었습니다	`30sec`
`silenceAutosupport`	AutoSupport 번들을 NetApp로 보내지 마십시오 자동으로 계층화	`false`
`autosupportImage`	AutoSupport 텔레메트리 컨테이너 이미지입니다	"`netapp/trident-autosupport:24.02`"
`autosupportProxy`	AutoSupport를 보내는 프록시의 주소/포트입니다 원격 측정	"`http://proxy.example.com:8888`"
`uninstall`	Astra Trident를 제거하는 데 사용되는 플래그입니다	`false`
`logFormat`	사용할 Astra Trident 로깅 형식[text,json]	"`text`"
`tridentImage`	설치할 Astra Trident 이미지	"`netapp/trident:24.02`"
`imageRegistry`	형식의 내부 레지스트리 경로입니다 `<registry FQDN>[:port][/subpath]`	"`k8s.gcr.io/sig-storage`" (Kubernetes 1.19 이상) 또는 "`quay.io/k8scsi`"
`kubeletDir`	호스트의 kubelet 디렉토리에 대한 경로입니다	"`/var/lib/kubelet`"
`wipeout`	의 전체 제거를 수행하기 위해 삭제할 리소스 목록입니다 아스트라 트리덴트	
`imagePullSecrets`	내부 레지스트리에서 이미지를 가져올 수 있는 비밀	
`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`

매개 변수	설명	기본값
`controllerPluginNodeSelector`	Pod용 추가 노드 선택기 는 와 동일한 형식을 따릅니다 `pod.spec.nodeSelector`.	기본값 없음, 선택 사항
`controllerPluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 는 와 동일한 형식을 따릅니다 `pod.spec.Tolerations`.	기본값 없음, 선택 사항
`nodePluginNodeSelector`	Pod용 추가 노드 선택기 는 와 동일한 형식을 따릅니다 `pod.spec.nodeSelector`.	기본값 없음, 선택 사항
`nodePluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 는 와 동일한 형식을 따릅니다 `pod.spec.Tolerations`.	기본값 없음, 선택 사항

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

강제 분리에 대한 세부 정보

에 대해 강제 분리를 사용할 수 있습니다 `ontap-san` 및 `ontap-san-economy` 만 해당. 강제 분리를 활성화하기 전에 Kubernetes 클러스터에서 비정상 노드 종료(ngns)를 활성화해야 합니다. 자세한 내용은 을 참조하십시오 "Kubernetes: 노드 정상 종료 아님".

Astra Trident는 Kubernetes ngns에 의존하므로 제거하지 마십시오 `out-of-service` 허용 불가능한 모든 작업 부하가 재조정될 때까지 상태가 불량한 노드에서 오인합니다. 무모하게 타트를 적용하거나 제거하면 백엔드 데이터 보호가 위태롭게 될 수 있습니다.

Kubernetes 클러스터 관리자가 를 적용했을 때 `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` 노드 및 에 대한 태트 `enableForceDetach` 가 로 설정되어 있습니다 `true`, Astra Trident에서 노드 상태를 확인하고 다음을 수행합니다.

1. 해당 노드에 마운트된 볼륨에 대한 백엔드 입출력 액세스를 중단합니다.
2. Astra Trident 노드 객체를 로 표시합니다 `dirty` (새 발행물에는 안전하지 않음).

Trident 컨트롤러는 노드가 재검증될 때까지(로 표시된 후) 새로운 게시 볼륨 요청을 거부합니다 `dirty`) Trident 노드 POD를 사용합니다. 클러스터 노드가 정상 상태가 되고 준비된 후에도 마운트된 PVC로 예약된 워크로드는 Astra Trident가 노드를 확인할 때까지 허용되지 않습니다 `clean` (새 출판물에 대해 안전).

노드 상태가 복원되고 Tint가 제거되면 Astra Trident는 다음을 수행합니다.

1. 노드에서 오래된 게시된 경로를 식별하고 제거합니다.
2. 노드가 에 있는 경우 `cleanable` 상태(Out-of-service taint가 제거되었으며 노드가 IN 상태입니다 `Ready` State) 및 게시된 모든 경로가 깨끗하며, Astra Trident가 노드를 다시 입원 처리하는 역할을 합니다 `clean` 노드에 게시된 새 볼륨을 허용합니다.

샘플 구성

에서 속성을 사용할 수 있습니다 구성 옵션 정의할 때 `TridentOrchestrator` 를 눌러 설치를 사용자 정의합니다.

기본 사용자 정의 구성

다음은 기본 사용자 정의 설치의 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
```

노드 선택기

이 예에서는 노드 선택기와 함께 Astra Trident를 설치합니다.

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

Windows 작업자 노드

이 예에서는 Windows 작업자 노드에 Astra Trident를 설치합니다.

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

AKS 클러스터에서 관리되는 **ID**입니다

이 예에서는 Astra Trident를 설치하여 AKS 클러스터에서 관리되는 ID를 활성화합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
```

AKS 클러스터에서 클라우드 **ID**입니다

이 예에서는 AKS 클러스터에서 클라우드 ID와 함께 사용할 Astra Trident를 설치합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx'
```

EKS 클러스터에서 클라우드 **ID**입니다

이 예에서는 AKS 클러스터에서 클라우드 ID와 함께 사용할 Astra Trident를 설치합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/astratrident-role'"
```