SnapMirror를 사용하여 볼륨을 복제합니다

Astra Trident

NetApp
July 18, 2025

목차

SnapMirror를 사용하여 볼륨을 복제합니다

Astra Control Provisioner를 사용하면 재해 복구를 위한 데이터 복제를 위해 한 클러스터의 소스 볼륨과 피어링된 클러스터의 타겟 볼륨 간에 미러 관계를 생성할 수 있습니다. 이름이 지정된 CRD(사용자 지정 리소스 정의)를 사용하여 다음 작업을 수행할 수 있습니다.

- 볼륨 간 미러 관계 생성(PVC)
- 볼륨 간 미러 관계를 제거합니다
- 미러 관계를 해제합니다
- 재해 상태(페일오버) 중에 2차 볼륨 승격
- 계획된 페일오버 또는 마이그레이션 중에 클러스터에서 클러스터로 애플리케이션의 무손실 전환 수행

복제 사전 요구 사항

시작하기 전에 다음과 같은 사전 요구 사항이 충족되는지 확인하십시오.

ONTAP 클러스터

- * Astra Control Provisioner *: Astra Control Provisioner 버전 23.10 이상이 ONTAP를 백엔드로 사용하는 소스 및 대상 Kubernetes 클러스터 모두에 있어야 합니다.
- * 라이센스 *: 소스 및 대상 ONTAP 클러스터 모두에서 데이터 보호 번들을 사용하는 ONTAP SnapMirror 비동기 라이센스를 활성화해야 합니다. 자세한 내용은 을 "ONTAP의 SnapMirror 라이센스 개요" 참조하십시오.

피어링

- * 클러스터 및 SVM *: ONTAP 스토리지 백엔드를 피어링해야 합니다. 자세한 내용은 을 "클러스터 및 SVM 피어링 개요" 참조하십시오.

  두 ONTAP 클러스터 간의 복제 관계에 사용되는 SVM 이름이 고유한지 확인합니다.

- * Astra Control Provisioner 및 SVM *: 피어링된 원격 SVM을 대상 클러스터의 Astra Control Provisioner에서 사용할 수 있어야 합니다.

지원되는 드라이버

- 볼륨 복제는 ONTAP-NAS 및 ONTAP-SAN 드라이버에서 지원됩니다.

대칭 복사된 PVC를 작성합니다

다음 단계를 수행하고 CRD 예제를 사용하여 운영 볼륨과 보조 볼륨 간의 미러 관계를 생성합니다.

단계

1. 운영 Kubernetes 클러스터에서 다음 단계를 수행합니다.
   a. 매개 변수를 사용하여 StorageClass 개체를 `trident.netapp.io/replication: true` 만듭니다.

예

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "nfs"
  trident.netapp.io/replication: "true"
```

b. 이전에 생성된 StorageClass를 사용하여 PVC를 생성합니다.

예

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

c. 로컬 정보로 MirrorRelationship CR을 생성합니다.

예

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Astra Control Provisioner는 볼륨과 볼륨의 현재 DP(Data Protection) 상태에 대한 내부 정보를 가져온 다음 MirrorRelationship의 상태 필드를 채웁니다.

d. TridentMirrorRelationship CR을 가져와 PVC의 내부 이름과 SVM을 얻습니다.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. 보조 Kubernetes 클러스터에서 다음 단계를 수행합니다.
   a. trident.netapp.io/replication: true 매개 변수를 사용하여 StorageClass 를 만듭니다.

      예

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        trident.netapp.io/replication: true
      ```

   b. 대상 및 소스 정보를 사용하여 MirrorRelationship CR을 생성합니다.

예

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Astra Control Provisioner는 구성된 관계 정책 이름(또는 ONTAP의 기본값)을 사용하여 SnapMirror 관계를 생성하고 초기화합니다.

c. 이전에 생성한 StorageClass를 사용하여 PVC를 생성하여 보조(SnapMirror 대상) 역할을 합니다.

예

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Astra Control Provisioner가 TridentMirrorRelationship CRD를 확인하고 관계가 없는 경우 볼륨을 생성하지 못합니다. 이 관계가 있으면 Astra Control Provisioner는 새로운 FlexVol 볼륨을 MirrorRelationship에 정의된 원격 SVM과 함께 피어링된 SVM에 배치합니다.

볼륨 복제 상태입니다

Trident Mirror Relationship(TMR)은 PVC 간 복제 관계의 한쪽 끝을 나타내는 CRD입니다. 대상 TMR에는 Astra Control Provisioner에게 원하는 상태를 알려주는 상태가 있습니다. 대상 TMR의 상태는 다음과 같습니다.

- * 설립 * : 로컬 PVC는 미러 관계의 대상 볼륨이며, 이것은 새로운 관계입니다.
- * 승진된 * : 로컬 PVC는 현재 유효한 미러 관계가 없는 ReadWrite 및 마운트 가능합니다.

- * 재설립 * : 로컬 PVC는 미러 관계의 대상 볼륨이며 이전에 해당 미러 관계에 있었습니다.
  - 대상 볼륨이 대상 볼륨 내용을 덮어쓰므로 대상 볼륨이 소스 볼륨과 관계가 있는 경우 다시 설정된 상태를 사용해야 합니다.
  - 볼륨이 소스와 이전에 관계가 없는 경우 재설정된 상태가 실패합니다.

비계획 페일오버 중에 보조 **PVC**를 승격합니다

보조 Kubernetes 클러스터에서 다음 단계를 수행합니다.

- TridentMirrorRelationship의 _spec.state_field를 로 `promoted`업데이트합니다.

계획된 페일오버 중에 보조 **PVC**를 승격합니다

계획된 장애 조치(마이그레이션) 중에 다음 단계를 수행하여 보조 PVC를 승격합니다.

단계

1. 운영 Kubernetes 클러스터에서 PVC의 스냅샷을 생성하고 스냅샷이 생성될 때까지 기다립니다.
2. 운영 Kubernetes 클러스터에서 SnapshotInfo CR을 생성하여 내부 세부 정보를 가져옵니다.

   예

   ```
   kind: SnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshot-name: csi-nas-snapshot
   ```

3. 보조 Kubernetes 클러스터에서 *TridentMirrorRelationship_CR* 의 *_spec.state_field*를 *_promitted* 및 *spec.promotedSnapshotHandle* 으로 업데이트하여 스냅샷의 내부 이름으로 업데이트합니다.
4. 보조 Kubernetes 클러스터에서 승격될 TridentMirrorRelationship의 상태(status.state 필드)를 확인합니다.

페일오버 후 미러 관계를 복구합니다

미러 관계를 복구하기 전에 새 1차 사이트로 만들 측면을 선택합니다.

단계

1. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 _spec.remoteVolumeHandle_field 값이 업데이트되었는지 확인합니다.
2. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 _spec.mirror_field를 로 `reestablished`업데이트합니다.

추가 작업

Astra Control Provisioner는 운영 볼륨과 2차 볼륨에서 다음 작업을 지원합니다.

1차 PVC를 새로운 2차 PVC로 복제합니다

이미 1차 PVC와 2차 PVC가 있는지 확인하십시오.

단계

1. 설정된 보조(대상) 클러스터에서 PersistentVolumeClaim 및 TridentMirrorRelationship CRD를 삭제합니다.
2. 운영(소스) 클러스터에서 TridentMirrorRelationship CRD를 삭제합니다.
3. 설정하려는 새 2차(대상) PVC에 대해 1차(소스) 클러스터에 새 TridentMirrorRelationship CRD를 생성합니다.

대칭 복사, 1차 또는 2차 PVC의 크기를 조정합니다

PVC는 평소대로 크기를 조정할 수 있으며, 데이터 양이 현재 크기를 초과할 경우 ONTAP는 자동으로 대상 flevxols를 확장합니다.

PVC에서 복제를 제거합니다

복제를 제거하려면 현재 보조 볼륨에 대해 다음 작업 중 하나를 수행합니다.

- 2차 PVC에서 MirrorRelationship을 삭제합니다. 이렇게 하면 복제 관계가 끊어집니다.
- 또는 spec.state 필드를 _promessed_로 업데이트합니다.

PVC 삭제(이전에 미러링됨)

Astra Control Provisioner는 복제된 PVC를 확인하고 볼륨을 삭제하기 전에 복제 관계를 해제합니다.

TMR을 삭제합니다

미러링된 관계의 한 쪽에서 TMR을 삭제하면 Astra Control Provisioner가 삭제를 완료하기 전에 나머지 TMR이 _promessed_state로 전환됩니다. 삭제하도록 선택한 TMR이 이미 _PROJED_STATE에 있는 경우 기존 미러 관계가 없으며 TMR이 제거되고 Astra Control Provisioner가 로컬 PVC를 _ReadWrite_로 승격합니다. 이렇게 삭제하면 ONTAP의 로컬 볼륨에 대한 SnapMirror 메타데이터가 해제됩니다. 이 볼륨이 향후 미러 관계에 사용될 경우 새 미러 관계를 생성할 때 _establed_volume 복제 상태의 새 TMR을 사용해야 합니다.

ONTAP가 온라인 상태일 때 미러 관계를 업데이트합니다

미러 관계는 설정된 후 언제든지 업데이트할 수 있습니다. 또는 필드를 사용하여 관계를 업데이트할 수 `state: promoted` `state: reestablished` 있습니다. 대상 볼륨을 일반 ReadWrite 볼륨으로 승격할 때 _promotedSnapshotHandle_을 사용하여 현재 볼륨을 복구할 특정 스냅샷을 지정할 수 있습니다.

ONTAP이 오프라인일 때 미러 관계를 업데이트합니다

Astra Control이 ONTAP 클러스터에 직접 연결되지 않은 상태에서 CRD를 사용하여 SnapMirror 업데이트를 수행할 수 있습니다. 다음 TridentActionMirrorUpdate 예제 형식을 참조하십시오.

예

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` TridentActionMirrorUpdate CRD의 상태를 반영합니다. 이 값은 *SUCCEEDED*,*In Progress* 또는_Failed_에서 가져올 수 있습니다.

Astra Control Provisioner를 활성화합니다

Trident 버전 23.10 이상에는 라이센스가 있는 Astra Control 사용자가 고급 스토리지 프로비저닝 기능에 액세스할 수 있도록 Astra Control Provisioner를 사용하는 옵션이 포함되어 있습니다. Astra Control Provisioner는 표준 Astra Trident CSI 기반 기능과 더불어 이 확장 기능을 제공합니다. 이 절차를 사용하여 Astra Control Provisioner를 활성화하고 설치할 수 있습니다.

Astra Control Service 구독에는 Astra Control Provisioner 사용에 대한 라이센스가 자동으로 포함됩니다.

향후 Astra Control 업데이트에서 Astra Control Provisioner는 Astra Trident를 스토리지 프로비저닝 및 오케스트레이터로 대체하며 Astra Control을 사용하려면 필수입니다. 따라서 Astra Control 사용자가 Astra Control Provisioner를 활성화하는 것이 좋습니다. Astra Trident는 오픈 소스를 계속 유지하며, NetApp의 새로운 CSI 및 기타 기능으로 릴리즈, 유지, 지원 및 업데이트될 것입니다.

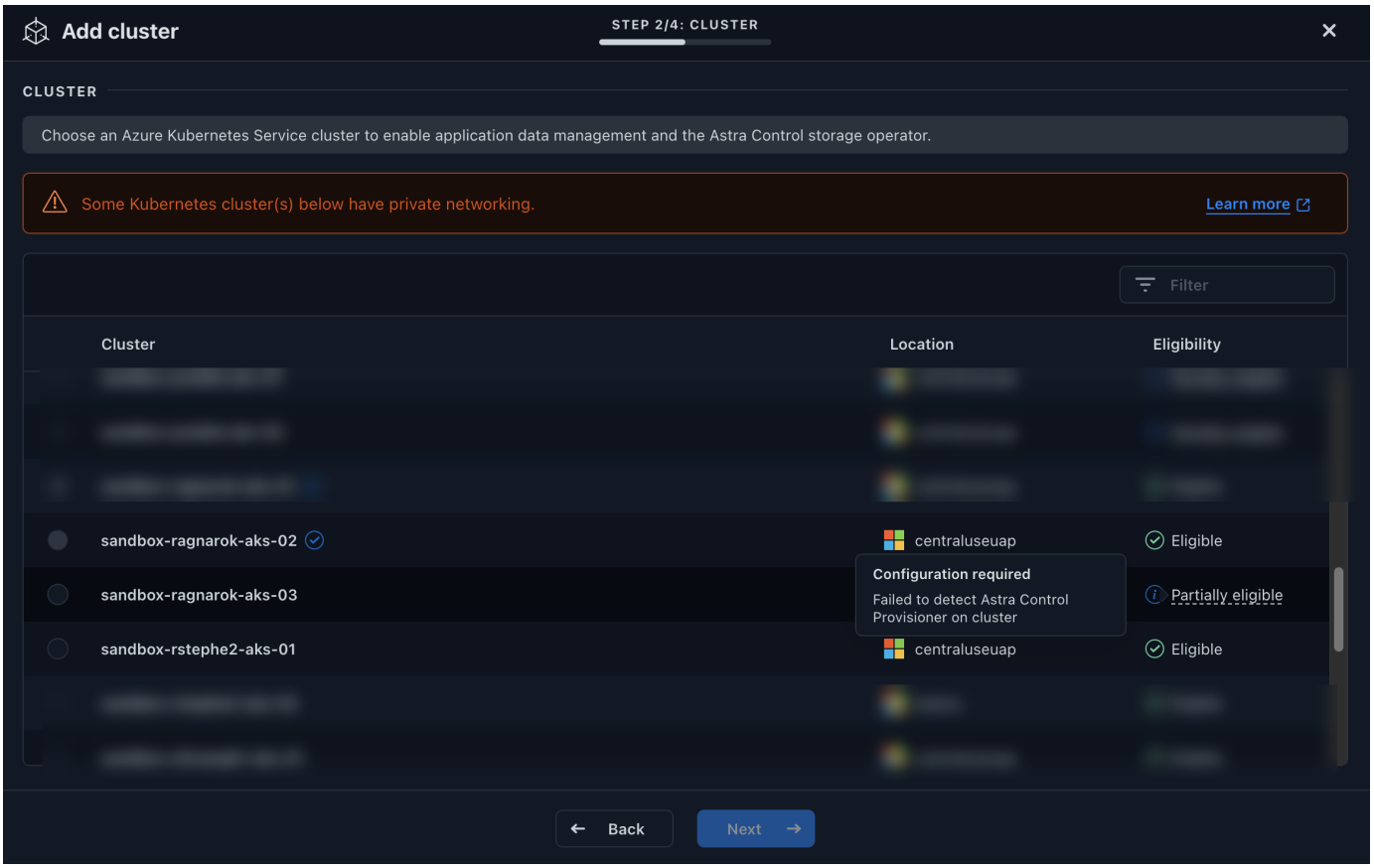
Astra Control Provisioner를 활성화해야 하는지 어떻게 알 수 있습니까**?**

이전에 Astra Trident를 설치하지 않은 클러스터를 Astra Control Service에 추가하면 클러스터가 로 표시됩니다 `Eligible`. 이후에는 "클러스터를 Astra Control에 추가합니다"Astra Control Provisioner가 자동으로 활성화됩니다.

클러스터가 표시되어 있지 않으면 `Eligible`다음 중 하나로 인해 표시됩니다 `Partially eligible`.

- 이전 버전의 Astra Trident를 사용하고 있습니다
- Provisioner 옵션이 아직 활성화되지 않은 Astra Trident 23.10을 사용하고 있습니다
- 자동 활성화를 허용하지 않는 클러스터 유형입니다

> `Partially eligible`경우에 따라 다음 지침을 사용하여 클러스터에 `Astra Control Provisioner`를 수동으로 활성화하십시오.



Astra Control Provisioner를 활성화하기 전에

Astra Control Provisioner가 없는 기존 Astra Trident가 있고 Astra Control Provisioner를 활성화하려면 먼저 다음을 수행합니다.

- * Astra Trident를 설치한 경우 해당 버전이 4개의 릴리즈 창 내에 있는지 확인 *: Astra Trident가 버전 24.02의 4개의 릴리즈 창 내에 있는 경우 Astra Control Provisioner를 사용하여 Astra Trident 24.02로 직접 업그레이드할 수 있습니다. 예를 들어, Astra Trident 23.04에서 24.02로 직접 업그레이드할 수 있습니다.
- * 클러스터에 AMD64 시스템 아키텍처가 있는지 확인 *: Astra Control Provisioner 이미지는 AMD64 및 ARM64 CPU 아키텍처 모두에서 제공되지만 Astra Control에서는 AMD64만 지원됩니다.

단계

1. NetApp Astra Control 이미지 레지스트리에 액세스:
   a. Astra Control Service UI에 로그온하고 Astra Control 계정 ID를 기록합니다.
      i. 페이지 오른쪽 상단의 그림 아이콘을 선택합니다.
      ii. API 액세스 * 를 선택합니다.
      iii. 계정 ID를 기록합니다.
   b. 같은 페이지에서 * API 토큰 생성 * 을 선택하고 API 토큰 문자열을 클립보드에 복사하여 편집기에 저장합니다.
   c. 원하는 방법을 사용하여 Astra Control 레지스트리에 로그인합니다.

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

```
crane auth login cr.astra.netapp.io -u <account-id> -p <api-token>
```

2. (사용자 지정 레지스트리에만 해당) 이미지를 사용자 지정 레지스트리로 이동하려면 다음 단계를 수행하십시오. 레지스트리를 사용하지 않는 경우 의 Trident 운영자 단계를 다음 섹션을 참조하십시오따르십시오.

> 다음 명령에 Docker 대신 Podman을 사용할 수 있습니다. Windows 환경을 사용하는 경우 PowerShell을 사용하는 것이 좋습니다.

Docker 를 참조하십시오

a. 레지스트리에서 Astra Control Provisioner 이미지를 가져옵니다.

> 가져온 이미지는 여러 플랫폼을 지원하지 않으며 Linux AMD64와 같이 이미지를 가져온 호스트와 동일한 플랫폼만 지원합니다.

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0
--platform <cluster platform>
```

예:

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0
--platform linux/amd64
```

b. 이미지에 태그 지정:

```
docker tag cr.astra.netapp.io/astra/trident-acp:24.02.0
<my_custom_registry>/trident-acp:24.02.0
```

c. 이미지를 사용자 지정 레지스트리에 푸시합니다.

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

크레인

a. Astra Control Provisioner 매니페스트를 사용자 지정 레지스트리에 복사합니다.

```
crane copy cr.astra.netapp.io/astra/trident-acp:24.02.0
<my_custom_registry>/trident-acp:24.02.0
```

3. 원래의 Astra Trident 설치 방법에 가 있는지 확인합니다.
4. 원래 사용한 설치 방법을 사용하여 Astra Trident에서 Astra Control Provisioner를 활성화합니다.

Astra Trident 운영자

a. "Astra Trident 설치 프로그램을 다운로드하여 압축을 풉니다"..

b. Astra Trident를 아직 설치하지 않았거나 원본 Astra Trident 구축에서 연산자를 제거한 경우 다음 단계를 완료하십시오.

i. CRD 생성:

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.1
6.yaml
```

ii. 트리덴트 이름 공간을 (`kubectl create namespace trident`생성하거나 트리덴트 이름 공간이 여전히 존재하는지 (`kubectl get all -n trident`확인합니다. 네임스페이스가 제거된 경우 다시 만듭니다.

c. Astra Trident를 24.02.0으로 업데이트:

Kubernetes 1.24 이하를 실행하는 클러스터의 경우 를 `bundle_pre_1_25.yaml`사용합니다. Kubernetes 1.25 이상을 실행하는 클러스터의 경우 를 `bundle_post_1_25.yaml`사용합니다.

```
kubectl -n trident apply -f trident-installer/deploy/<bundle-
name.yaml>
```

d. Astra Trident가 실행 중인지 확인합니다.

```
kubectl get torc -n trident
```

응답:

```
NAME      AGE
trident   21m
```

e.] 비밀을 사용하는 레지스트리가 있는 경우 Astra Control Provisioner 이미지를 가져오는 데 사용할 비밀을 만듭니다.

```
kubectl create secret docker-registry <secret_name> -n trident
--docker-server=<my_custom_registry> --docker-username=<username>
--docker-password=<token>
```

f. TridentOrchestrator CR을 편집하고 다음과 같이 편집합니다.

```
kubectl edit torc trident -n trident
```

i. Astra Trident 이미지에 대한 사용자 지정 레지스트리 위치를 설정하거나 Astra Control 레지스트리 또는 에서 가져옵니다 (`tridentImage: <my_custom_registry>/trident:24.02.0` `tridentImage: netapp/trident:24.02.0`.

ii. Astra Control Provisioner 사용 (`enableACP: true`).

iii. Astra Control Provisioner 이미지의 사용자 지정 레지스트리 위치를 설정하거나 Astra Control 레지스트리 또는 에서 (`acpImage: <my_custom_registry>/trident-acp:24.02.0` `acpImage: cr.astra.netapp.io/astra/trident-acp:24.02.0`가져옵니다.

iv. 이 절차의 앞부분에서 설정한 경우 이미지 풀 암호 여기에서 설정할 수 (`imagePullSecrets: - <secret_name>`있습니다.) 이전 단계에서 설정한 것과 동일한 이름 암호 이름을 사용합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  tridentImage: <registry>/trident:24.02.0
  enableACP: true
  acpImage: <registry>/trident-acp:24.02.0
  imagePullSecrets:
  - <secret_name>
```

g. 파일을 저장하고 종료합니다. 배포 프로세스가 자동으로 시작됩니다.

h. 운영자, 배포 및 복제 세트가 생성되었는지 확인합니다.

```
kubectl get all -n trident
```

Kubernetes 클러스터에는 운영자의 인스턴스 * 하나가 있어야 합니다. Astra Trident 연산자를 여러 번 구축해서는 안 됩니다.

i. 컨테이너가 실행 중이고 의 상태가 다음과 같은지 확인합니다 `trident-acp` `acpVersion` `24.02.0` `Installed`.

```
kubectl get torc -o yaml
```

응답:

```
status:
  acpVersion: 24.02.0
  currentInstallationParams:
    ...
    acpImage: <registry>/trident-acp:24.02.0
    enableACP: "true"
    ...
  ...
  status: Installed
```

tridentctl 을 선택합니다

a. "Astra Trident 설치 프로그램을 다운로드하여 압축을 풉니다"..

b. "기존 Astra Trident가 있는 경우 이를 호스팅하는 클러스터에서 제거합니다"..

c. Astra Control Provisioner를 활성화한 상태로 Astra Trident 설치 (`--enable-acp=true`):

```
./tridentctl -n trident install --enable-acp=true --acp
-image=mycustomregistry/trident-acp:24.02
```

d. Astra Control Provisioner가 활성화되었는지 확인합니다.

```
./tridentctl -n trident version
```

응답:

```
+----------------+----------------+-------------+ | SERVER
VERSION | CLIENT VERSION | ACP VERSION | +----------------
+----------------+-------------+ | 24.02.0 | 24.02.0 | 24.02.0. |
+----------------+----------------+-------------+
```

헬름

a. Astra Trident 23.07.1 이전 버전을 설치한 경우 "설치 제거" 운영자 및 기타 구성 요소가 필요합니다.

b. Kubernetes 클러스터에서 1.24 이전 버전을 실행 중인 경우 psp:

```
kubectl delete psp tridentoperatorpod
```

c. Astra Trident Helm 리포지토리를 추가합니다.

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

d. 제어 차트 업데이트:

```
helm repo update netapp-trident
```

응답:

```
Hang tight while we grab the latest from your chart
repositories...
...Successfully got an update from the "netapp-trident" chart
repository
Update Complete. □Happy Helming!□
```

e. 영상을 나열합니다.

```
./tridentctl images -n trident
```

응답:

```
| v1.28.0            | netapp/trident:24.02.0|
|                    | docker.io/netapp/trident-
autosupport:24.02|
|                    | registry.k8s.io/sig-storage/csi-
provisioner:v4.0.0|
|                    | registry.k8s.io/sig-storage/csi-
attacher:v4.5.0|
|                    | registry.k8s.io/sig-storage/csi-
resizer:v1.9.3|
|                    | registry.k8s.io/sig-storage/csi-
snapshotter:v6.3.3|
|                    | registry.k8s.io/sig-storage/csi-node-
driver-registrar:v2.10.0 |
|                    | netapp/trident-operator:24.02.0 (optional)
```

f. 트라이덴트 - 운전자 24.02.0을 사용할 수 있는지 확인합니다.

```
helm search repo netapp-trident/trident-operator --versions
```

응답:

```
NAME                                CHART VERSION   APP VERSION
DESCRIPTION
netapp-trident/trident-operator 100.2402.0      24.02.0         A
```

g. `helm install`다음 설정을 포함하는 옵션 중 하나를 사용하고 실행합니다.

- 배포 위치의 이름입니다
- Astra Trident 버전
- Astra Control Provisioner 이미지의 이름
- Provisioner를 활성화하는 플래그입니다
- (선택 사항) 로컬 레지스트리 경로입니다. 로컬 레지스트리를 사용하는 경우 하나의 레지스트리 또는 다른 레지스트리에 을 "Trident 이미지" 배치할 수 있지만 모든 CSI 이미지는 동일한 레지스트리에 있어야 합니다.
- Trident 네임스페이스

옵션

◦ 레지스트리가 없는 이미지

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=cr.astra.netapp.io/astra/trident-
acp:24.02.0 --set enableACP=true --set operatorImage=netapp/trident-
operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

◦ 하나 이상의 레지스트리에 있는 이미지

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=<your-registry>:<acp image> --set
enableACP=true --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=netapp/trident-operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

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

Helm을 사용하여 Trident를 구축하는 데 문제가 있는 경우 다음 명령을 실행하여 Astra Trident를 완전히 제거합니다.

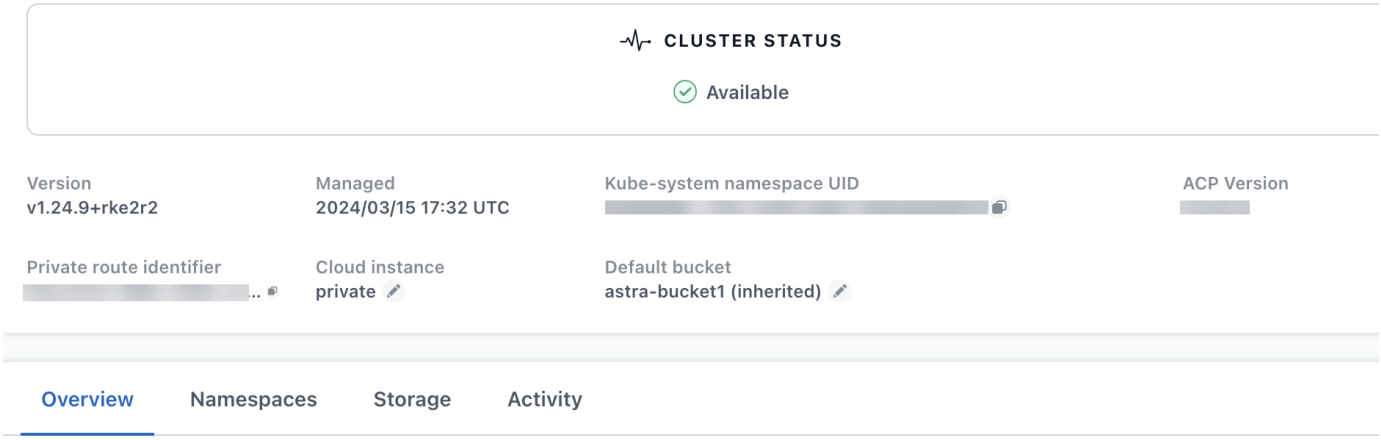
```
./tridentctl uninstall -n trident
```

- Astra Control Provisioner를 다시 활성화하기 전에 설치 제거의 일부로 사용하지 마십시오 "Astra Trident CRD를 완전히 제거합니다" .

결과

Astra Control Provisioner 기능이 활성화되어 있으며 실행 중인 버전에 제공되는 모든 기능을 사용할 수 있습니다.

Astra Control Provisioner를 설치한 후 Astra Control UI에서 Provisioner를 호스팅하는 클러스터에 `ACP version` 필드와 현재 설치된 버전 번호가 아닌 대신 `Trident version` 표시됩니다.



를 참조하십시오

- "Astra Trident 업그레이드 설명서"