Astra Trident 관리

Astra Trident

NetApp
November 14, 2025

This PDF was generated from https://docs.netapp.com/ko-kr/trident-2304/trident-managing-k8s/upgrade-trident.html on November 14, 2025. Always check docs.netapp.com for the latest.

목차

Astra Trident 관리 . . . 1
- Astra Trident를 업그레이드합니다 . . . 1
  - Astra Trident를 업그레이드합니다 . . . 1
  - 운영자와 함께 업그레이드하십시오 . . . 3
  - tridentctl로 업그레이드하십시오 . . . 11
- Astra Trident를 제거합니다 . . . 14
  - Helm을 사용하여 제거합니다 . . . 14
  - Trident 연산자를 사용하여 제거합니다 . . . 15
  - 을 사용하여 를 제거합니다 `tridentctl` . . . 16
- Astra Trident를 다운그레이드하십시오 . . . 16
  - 다운그레이드 시점 . . . 16
  - 다운그레이드를 하지 않는 경우 . . . 16
  - 운영자를 통해 Astra Trident가 설치된 경우의 다운그레이드 프로세스 . . . 16
  - 을 사용하여 Astra Trident를 설치한 경우의 다운그레이드 프로세스 `tridentctl` . . . 18

Astra Trident 관리

Astra Trident를 업그레이드합니다

Astra Trident를 업그레이드합니다

Astra Trident는 분기별 릴리스 케이던스를 따르며, 매년 4개의 주요 릴리즈를 제공합니다. 각 새로운 릴리스는 이전 릴리즈를 기반으로 하며 새로운 기능과 성능 향상, 버그 수정 및 개선 기능을 제공합니다. Astra Trident의 새로운 기능을 이용하려면 1년에 한 번 이상 업그레이드하시기 바랍니다.

업그레이드 전 고려 사항

Astra Trident의 최신 릴리즈로 업그레이드할 때 다음 사항을 고려하십시오.

- 주어진 Kubernetes 클러스터의 모든 네임스페이스에 하나의 Astra Trident 인스턴스만 설치되어야 합니다.
- Trident 20.01부터 베타 릴리즈만 제공됩니다 "볼륨 스냅숏" 가 지원됩니다. Kubernetes 관리자는 알파 스냅샷 개체를 베타로 안전하게 백업하거나 변환하여 레거시 알파 스냅샷을 유지하도록 주의해야 합니다.
  - CSI 볼륨 스냅샷은 이제 Kubernetes 1.20부터 시작되는 GA 기능입니다. 업그레이드하기 전에 를 사용하여 알파 스냅샷 CRD를 제거해야 합니다 `tridentctl obliviate alpha-snapshot-crd` 알파 스냅샷 사양에 대한 CRD를 삭제합니다.
  - 볼륨 스냅샷의 베타 릴리스에는 수정된 CRD 세트와 스냅샷 컨트롤러가 도입되며, 이 두 가지는 모두 Astra Trident를 업그레이드하기 전에 설정해야 합니다.
  - 자세한 내용은 을 참조하십시오 "Kubernetes 클러스터를 업그레이드하기 전에 알아야 할 사항".
- 버전 19.04 및 이전 버전에서 업그레이드하는 경우 IT에서 Astra Trident 메타데이터를 마이그레이션해야 합니다 `etcd` CRD 개체. 를 확인하십시오 "Astra Trident 릴리즈별 문서" 업그레이드 작동 방식을 이해합니다.
- 업그레이드할 때 제공하는 것이 중요합니다 `parameter.fsType` 인치 `StorageClasses` Astra Trident에서 사용 삭제하고 다시 만들 수 있습니다 `StorageClasses` 기존 볼륨을 그대로 사용합니다.
  - 이것은 시행에 대한 ** 요구 사항입니다 "보안 컨텍스트" SAN 볼륨:
  - sample input 디렉토리에는 https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-basic.yaml.templ와 같은 예가 포함되어 있습니다[`storage-class-basic.yaml.templ`^] 및 링크: `storage-class-bronze-default.yaml`. 자세한 내용은 을 참조하십시오 "알려진 문제".

1단계**:** 버전을 선택합니다

Astra Trident 버전은 날짜를 기반으로 합니다 `YY.MM` 이름 지정 규칙. 여기서 "YY"는 연도의 마지막 두 자리이고 "MM"은 월입니다. DOT 릴리스는 a를 따릅니다 `YY.MM.X` 규칙. 여기서 "X"는 패치 수준입니다. 업그레이드할 버전에 따라 업그레이드할 버전을 선택합니다.

- 설치된 버전의 4 릴리스 창 내에 있는 모든 대상 릴리스에 대해 직접 업그레이드를 수행할 수 있습니다. 예를 들어 22.04에서 23.04로 직접 업그레이드할 수 있습니다(22.04.1과 같은 도트 릴리스 포함).
- 이전 버전이 설치되어 있는 경우, 특정 지침은 해당 릴리스의 설명서를 사용하여 다단계 업그레이드를 수행해야 합니다. 따라서 먼저 4개의 릴리즈 창에 맞는 최신 릴리즈로 업그레이드해야 합니다. 예를 들어 18.07을 실행하고 20.07 릴리스로 업그레이드하려는 경우 다음과 같이 다단계 업그레이드 프로세스를 따르십시오.

a. 첫 번째 업그레이드는 18.07에서 19.07로 가능합니다.

b. 그런 다음 19.07에서 20.07로 업그레이드합니다.

> OpenShift Container Platform에서 Trident 연산자를 사용하여 업그레이드할 때는 Trident 21.01.1 이상으로 업그레이드해야 합니다. 21.01.0으로 릴리스된 Trident 연산자에는 21.01.1에서 해결된 알려진 문제가 포함되어 있습니다. 자세한 내용은 를 참조하십시오 "GitHub에 대한 발행 세부 정보".

2단계**:** 원래 설치 방법을 결정합니다

일반적으로 초기 설치에 사용한 것과 동일한 방법으로 업그레이드해야 하지만 "설치 방법 간에 이동합니다".

처음에 Astra Trident를 설치하는 데 사용한 버전을 확인하려면:

1. 사용 `kubectl get pods - trident` 를 눌러 포드를 검사합니다.
   - 운영자 POD가 없는 경우, 를 사용하여 Astra Trident를 설치했습니다 `tridentctl`.
   - 운영자 포드가 있는 경우, Trident 연산자를 사용하여 수동으로 또는 Hrom을 사용하여 Astra Trident를 설치했습니다.
2. 작업자 포드가 있는 경우 를 사용합니다 `kubectl describe tproc trident` Helm을 사용하여 Astra Trident가 설치되었는지 확인합니다.
   - H제어 레이블이 있는 경우, Hrom을 사용하여 Astra Trident를 설치했습니다.
   - H제어 레이블이 없는 경우 Trident 연산자를 사용하여 Astra Trident를 수동으로 설치했습니다.

3단계**:** 업그레이드 방법을 선택합니다

Astra Trident를 업그레이드하는 방법에는 두 가지가 있습니다.

운용자를 통한 업그레이드 시기

가능합니다 "Trident 연산자를 사용하여 업그레이드합니다" 조건:

- 처음에 운영자 또는 을(를) 사용하여 Astra Trident를 설치했습니다 `tridentctl`.
- CSI Trident를 제거해도 설치 시 메타데이터가 유지됩니다.
- CSI 기반 Astra Trident가 설치되어 있습니다. 의 19.07에서 모든 릴리스는 CSI 기반입니다. Trident 네임스페이스의 Pod를 검사하여 버전을 확인할 수 있습니다.
  - 23.01 이전 버전의 POD 이름 지정은 다음을 사용합니다. `trident-csi-*`
  - 23.01 이상에서 포드 이름 지정 시 사용:
    - `trident-controller-<generated id>` 컨트롤러 포드의 경우
    - `trident-node-<operating system>-<generated id>` 노드 Pod용
    - `trident-operator-<generated id>` 작업자 포드의 경우

> 를 사용하는 경우 연산자를 사용하여 Trident를 업그레이드하지 마십시오 `etcd`기반 Trident 릴리즈(19.04 이상)

를 사용하여 업그레이드해야 하는 경우 `tridentctl`

가능합니다 처음에 'tridenctl'을 사용하여 Astra Trident를 설치한 경우

`tridentctl` Astra Trident를 설치하는 일반적인 방법이며 복잡한 사용자 정의가 필요한 사용자에게 가장 많은 옵션을 제공합니다. 자세한 내용은 을 참조하십시오 "설치 방법을 선택합니다".

조작자에 대한 변경

Astra Trident의 21.01 릴리즈에는 운영자에게 구조적 변경 사항이 도입되었습니다.

- 이제 연산자가 * 클러스터 범위 * 가 됩니다. Trident 연산자(버전 20.04 - 20.10)의 이전 인스턴스는 * 네임스페이스 범위 * 였습니다. 클러스터 범위의 연산자는 다음과 같은 이유로 유용합니다.
  - 리소스 책임: 이제 운영자는 클러스터 수준에서 Astra Trident 설치와 관련된 리소스를 관리합니다. Astra Trident를 설치하는 과정에서 운영자는 를 사용하여 여러 리소스를 생성하고 유지 관리합니다 `ownerReferences`. 유지 관리 `ownerReferences` 클러스터 범위 리소스의 경우 OpenShift와 같은 특정 Kubernetes 배포판에서 오류가 발생할 수 있습니다. 이 문제는 클러스터 범위 운영자를 통해 완화됩니다. Trident 리소스의 자동 복구 및 패칭은 필수 요구사항입니다.
  - 제거 중 정리: Astra Trident를 완전히 제거하려면 모든 관련 리소스를 삭제해야 합니다. 네임스페이스 범위 연산자는 클러스터 범위 리소스(예: clusterRole, ClusterRoleBinding 및 PodSecurityPolicy)를 제거하는 데 문제가 있을 수 있으며 불완전한 정리 작업을 초래할 수 있습니다. 클러스터 범위 연산자로 인해 이 문제가 발생하지 않습니다. 사용자는 Astra Trident를 완전히 제거하고 필요한 경우 새로 설치할 수 있습니다.
- `TridentProvisioner` 이(가) 이제 로 대체됩니다 `TridentOrchestrator` Astra Trident를 설치 및 관리하는 데 사용되는 사용자 지정 리소스입니다. 또한 에 새 필드가 도입되었습니다 `TridentOrchestrator` 사양 사용자는 네임스페이스 Trident가 을 사용하여 설치/업그레이드되도록 지정할 수 있습니다 `spec.namespace` 필드에 입력합니다. 예를 들어 보겠습니다 "여기".

운영자와 함께 업그레이드하십시오

수동으로 또는 Hrom을 사용하여 기존 Astra Trident 설치를 간편하게 업그레이드할 수 있습니다.

Trident 연산자를 사용하여 업그레이드합니다

일반적으로 처음 설치하는 데 사용한 것과 동일한 방법으로 Astra Trident를 업그레이드해야 합니다. 검토 "업그레이드 방법을 선택합니다" Trident 연산자를 사용하여 업그레이드를 시도하기 전.

> Namespace 범위 연산자(버전 20.07 ~ 20.10)를 사용하여 설치된 Astra Trident의 인스턴스에서 업그레이드할 경우 Trident 연산자는 자동으로 다음을 수행합니다.
>
> - 마이그레이션 `tridentProvisioner` A로 `tridentOrchestrator` 같은 이름의 개체,
> - 삭제 `TridentProvisioner` 개체 및 `tridentprovisioner` CRD
> - Astra Trident를 사용 중인 클러스터 범위 운영자 버전으로 업그레이드합니다
> - 원래 설치된 곳에 Astra Trident 동일한 네임스페이스를 설치합니다

클러스터 범위 **Trident** 운영자 설치를 업그레이드합니다

클러스터 범위 Trident 운영자 설치를 업그레이드할 수 있습니다. Astra Trident 버전 21.01 이상에서는 클러스터 범위

연산자를 사용합니다.

시작하기 전에

실행 중인 Kubernetes 클러스터를 사용하고 있는지 확인합니다 "지원되는 Kubernetes 버전".

단계

1. Astra Trident 버전 확인:

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

2. 현재 Astra Trident 인스턴스를 설치하는 데 사용된 Trident 연산자를 삭제합니다. 예를 들어, 22.01에서 업그레이드하는 경우 다음 명령을 실행합니다.

```
kubectl delete -f 22.01/trident-installer/deploy/bundle.yaml -n trident
```

3. 를 사용하여 초기 설치를 사용자 지정한 경우 `TridentOrchestrator` 속성을 편집할 수 있습니다 `TridentOrchestrator` 설치 매개 변수를 수정하는 개체입니다. 여기에는 오프라인 모드에 대해 미러링된 Trident 및 CSI 이미지 레지스트리를 지정하는 변경 사항, 디버그 로그 활성화 또는 이미지 풀 비밀을 지정하는 변경 사항이 포함될 수 있습니다.
4. 사용자 환경과 Astra Trident 버전에 맞는 올바른 번들 YAML 파일을 사용하여 Astra Trident를 설치합니다. 예를 들어, Kubernetes 1.27용 Astra Trident 23.04를 설치하는 경우 다음 명령을 실행합니다.

```
kubectl create -f 23.04.0/trident-installer/deploy/bundle_post_1_25.yaml
-n trident
```

> Trident는 운영자를 설치하고 Kubernetes 버전용 관련 개체를 생성하는 데 사용할 수 있는 번들 파일을 제공합니다.
>
> - 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".

결과

Trident 운영자는 기존 Astra Trident 설치를 식별하고 운영자와 동일한 버전으로 업그레이드합니다.

네임스페이스 범위 연산자 설치를 업그레이드합니다

네임스페이스 범위 연산자(버전 20.07 ~ 20.10)를 사용하여 설치된 Astra Trident의 인스턴스에서 클러스터 범위 운영자 설치로 업그레이드할 수 있습니다.

시작하기 전에

네임스페이스 범위 운영자를 배포하는 데 사용되는 번들 YAML 파일이 필요합니다 `https://github.com/NetApp/trident/tree/stable/vXX.XX/deploy/BUNDLE.YAML` 위치 *`vXX.XX`*

은(는) 버전 번호입니다 *BUNDLE.YAML* 번들 YAML 파일 이름입니다.

단계

1. 를 확인합니다 `TridentProvisioner` 기존 Trident 설치의 상태는 입니다 `Installed`.

```
kubectl describe tprov trident -n trident | grep Message: -A 3

Message:  Trident installed
Status:   Installed
Version:  v20.10.1
```

상태가 표시되는 경우 `Updating`계속하기 전에 이 문제를 해결하십시오. 가능한 상태 값 목록은 를 참조하십시오 "여기".

2. 를 생성합니다 `TridentOrchestrator` Trident 설치 프로그램과 함께 제공된 매니페스트를 사용하여 CRD를 만듭니다.

```
# Download the release required [23.04.0]
mkdir 23.04.0
cd 23.04.0
wget
https://github.com/NetApp/trident/releases/download/v23.04.0/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. 매니페스트를 사용하여 네임스페이스 범위 연산자를 삭제합니다.
   a. 올바른 디렉토리에 있는지 확인하십시오.

```
pwd
/root/20.10.1/trident-installer
```

   b. 네임스페이스 범위 연산자를 삭제합니다.

```
kubectl delete -f deploy/<BUNDLE.YAML> -n trident

serviceaccount "trident-operator" deleted
clusterrole.rbac.authorization.k8s.io "trident-operator" deleted
clusterrolebinding.rbac.authorization.k8s.io "trident-operator"
deleted
deployment.apps "trident-operator" deleted
podsecuritypolicy.policy "tridentoperatorpods" deleted
```

c. Trident 운영자가 제거되었는지 확인합니다.

```
kubectl get all -n trident

NAME                                READY   STATUS    RESTARTS   AGE
pod/trident-csi-68d979fb85-dsrmn    6/6     Running   12         99d
pod/trident-csi-8jfhf               2/2     Running   6          105d
pod/trident-csi-jtnjz               2/2     Running   6          105d
pod/trident-csi-lcxvh               2/2     Running   8          105d

NAME                   TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)              AGE
service/trident-csi    ClusterIP   10.108.174.125   <none>
34571/TCP,9220/TCP   105d

NAME                           DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR                                        AGE
daemonset.apps/trident-csi   3         3         3       3
3           kubernetes.io/arch=amd64,kubernetes.io/os=linux   105d

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/trident-csi    1/1     1            1           105d

NAME                                      DESIRED   CURRENT   READY
AGE
replicaset.apps/trident-csi-68d979fb85    1         1         1
105d
```

4. (선택 사항) 설치 매개 변수를 수정해야 하는 경우 를 업데이트합니다 `TridentProvisioner` 사양 여기에는 의 값 변경과 같은 변경 사항이 포함될 수 있습니다 `tridentImage`, `autosupportImage`개인 이미지 저장소 및 제공 `imagePullSecrets`) 네임스페이스 범위 연산자를 삭제한 후 클러스터 범위 연산자를 설치하기 전에 먼저 . 업데이트할 수 있는 전체 매개 변수 목록은 을 참조하십시오 "구성 옵션".

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

5. Trident 클러스터 범위 연산자를 설치합니다.

   a. 올바른 디렉토리에 있는지 확인하십시오.

   ```
   pwd
   /root/23.04.0/trident-installer
   ```

   b. 클러스터 범위 연산자를 같은 네임스페이스에 설치합니다.

   > Trident는 운영자를 설치하고 Kubernetes 버전용 관련 개체를 생성하는 데 사용할 수 있는 번들 파일을 제공합니다.
   >
   > - 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".

   ```
   kubectl create -f deploy/<BUNDLE.YAML>

   serviceaccount/trident-operator created
   clusterrole.rbac.authorization.k8s.io/trident-operator created
   clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
   deployment.apps/trident-operator created
   podsecuritypolicy.policy/tridentoperatorpods created

   #All tridentProvisioners will be removed, including the CRD itself
   kubectl get tprov -n trident
   Error from server (NotFound): Unable to list "trident.netapp.io/v1,
   Resource=tridentprovisioners": the server could not find the
   requested resource (get tridentprovisioners.trident.netapp.io)

   #tridentProvisioners are replaced by tridentOrchestrator
   kubectl get torc
   NAME      AGE
   trident   13s
   ```

   c. 네임스페이스에서 Trident Pod를 검사합니다. 를 클릭합니다 `trident-controller` 및 POD 이름은 23.01에 도입된 명명 규칙을 반영합니다.

```
kubectl get pods -n trident

NAME                                  READY   STATUS    RESTARTS
AGE
trident-controller-79df798bdc-m79dc   6/6     Running   0
1m41s
trident-node-linux-xrst8              2/2     Running   0
1m41s
trident-operator-5574dbbc68-nthjv     1/1     Running   0
1m52s
```

d. Trident가 의도한 버전으로 업데이트되었는지 확인합니다.

```
kubectl describe torc trident | grep Message -A 3
Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v23.04.0
```

제어 기반 작업자 설치를 업그레이드합니다

제어 기반 운영자 설치를 업그레이드하려면 다음 단계를 수행하십시오.

> Astra Trident가 설치된 Kubernetes 클러스터를 1.24에서 1.25 이상으로 업그레이드할 경우 Values.YAML을 업데이트해야 합니다 `excludePodSecurityPolicy` 를 선택합니다 `true` 또는 을 추가합니다 `--set excludePodSecurityPolicy=true` 를 누릅니다 `helm upgrade` 명령을 먼저 실행한 후 클러스터를 업그레이드하십시오.

단계

1. 최신 Astra Trident 릴리스를 다운로드하십시오.
2. 를 사용합니다 `helm upgrade` 명령 위치 `trident-operator-23.04.0.tgz` 업그레이드하려는 버전을 반영합니다.

```
helm upgrade <name> trident-operator-23.04.0.tgz
```

초기 설치 중에 기본값이 아닌 옵션을 설정한 경우(예: Trident 및 CSI 이미지에 대한 전용, 미러 레지스트리 지정) 를 사용합니다 `--set` 이러한 옵션이 업그레이드 명령에 포함되도록 하려면 값이 기본값으로 재설정됩니다.

예를 들어, 의 기본값을 변경합니다 `tridentDebug`에서 다음 명령을 실행합니다.

```
helm upgrade <name> trident-operator-23.04.0-custom.tgz --set
tridentDebug=true
```

3. 실행 `helm list` 차트와 앱 버전이 모두 업그레이드되었는지 확인합니다. 실행 `tridentctl logs` 디버그 메시지를 검토합니다.

결과

Trident 운영자는 기존 Astra Trident 설치를 식별하고 운영자와 동일한 버전으로 업그레이드합니다.

비운영자 설치에서 업그레이드

에서 Trident 운영자의 최신 릴리즈로 업그레이드할 수 있습니다 `tridentctl` 설치:

단계

1. 최신 Astra Trident 릴리스를 다운로드하십시오.

```
# Download the release required [23.04.0]
mkdir 23.04.0
cd 23.04.0
wget
https://github.com/NetApp/trident/releases/download/v22.01.1/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
```

2. 를 생성합니다 `tridentorchestrator` 매니페스트에서 CRD를 선택합니다.

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

3. 클러스터 범위 연산자를 같은 네임스페이스에 구현합니다.

```
kubectl create -f deploy/<BUNDLE.YAML>

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                  READY   STATUS    RESTARTS   AGE
trident-controller-79df798bdc-m79dc   6/6     Running   0          150d
trident-node-linux-xrst8              2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv     1/1     Running   0          1m30s
```

4. 을 생성합니다 `TridentOrchestrator` Astra Trident 설치용 CR.

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

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s
```

5. Trident가 의도한 버전으로 업그레이드되었는지 확인합니다.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v23.04.0
```

결과

기존 백엔드 및 PVC는 자동으로 사용할 수 있습니다.

tridentctl로 업그레이드하십시오

을 사용하여 기존 Astra Trident 설치를 쉽게 업그레이드할 수 있습니다 `tridentctl`.

를 사용하여 **Astra Trident**를 업그레이드합니다 `tridentctl`

Astra Trident를 제거하고 다시 설치하면 업그레이드 역할을 합니다. Trident를 제거할 때 Astra Trident 배포에 사용되는 영구 볼륨 클레임(PVC) 및 영구 볼륨(PV)은 삭제되지 않습니다. 이미 프로비저닝된 PVS는 Astra Trident가 오프라인 상태인 동안 계속 사용할 수 있으며, Astra Trident는 다시 온라인 상태가 되면 중간 기간 동안 생성된 모든 PVC에 대해 볼륨을 프로비저닝합니다.

시작하기 전에

검토 "업그레이드 방법을 선택합니다" 를 사용하여 업그레이드하기 전에 `tridentctl`.

단계

1. 에서 제거 명령을 실행합니다 `tridentctl` CRD 및 관련 객체를 제외한 Astra Trident와 연결된 모든 리소스를 제거합니다.

```
./tridentctl uninstall -n <namespace>
```

2. Astra Trident를 다시 설치합니다. 을 참조하십시오 "tridentctl을 사용하여 Astra Trident를 설치합니다".

업그레이드 프로세스를 중단하지 마십시오. 설치 프로그램이 완료될 때까지 실행되는지 확인합니다.

를 사용하여 볼륨 업그레이드 `tridentctl`

업그레이드 후에는 새로운 Trident 릴리즈(예: 주문형 볼륨 스냅샷)에서 제공되는 다양한 기능을 사용하여 볼륨을 업그레이드할 수 있습니다 `tridentctl upgrade` 명령.

레거시 볼륨이 있는 경우 Astra Trident의 새로운 기능 세트를 사용하려면 NFS 또는 iSCSI 유형에서 CSI 유형으로 업그레이드해야 합니다. Trident에서 프로비저닝한 레거시 PV는 기존 기능 세트를 지원합니다.

시작하기 전에

볼륨을 CSI 유형으로 업그레이드하기 전에 다음 사항을 고려하십시오.

- 모든 볼륨을 업그레이드할 필요는 없습니다. 이전에 생성된 볼륨은 계속 액세스할 수 있으며 정상적으로 작동합니다.
- 업그레이드할 때 배포/StatefulSet 의 일부로 PV를 마운트할 수 있습니다. deployment/StatefulSet 을 아래로 가져올 필요는 없습니다.
- 업그레이드 시 독립 실행형 POD에 PV를 * 첨부할 수 없습니다. 볼륨을 업그레이드하기 전에 포드를 종료해야 합니다.
- PVC에 바인딩된 볼륨만 업그레이드할 수 있습니다. PVC에 바인딩되지 않은 용적은 업그레이드 전에 제거 및 가져와야 합니다.

단계

1. 실행 `kubectl get pv` PVS를 나열합니다.

```
kubectl get pv
NAME                        CAPACITY     ACCESS MODES   RECLAIM POLICY
STATUS   CLAIM                   STORAGECLASS    REASON   AGE
default-pvc-1-a8475         1073741824   RWO            Delete
Bound    default/pvc-1          standard                19h
default-pvc-2-a8486         1073741824   RWO            Delete
Bound    default/pvc-2          standard                19h
default-pvc-3-a849e         1073741824   RWO            Delete
Bound    default/pvc-3          standard                19h
default-pvc-4-a84de         1073741824   RWO            Delete
Bound    default/pvc-4          standard                19h
trident                     2Gi          RWO            Retain
Bound    trident/trident                                19h
```

현재 Trident 20.07에서 를 사용하여 생성한 PVS는 4개입니다 `netapp.io/trident` 공급자.

2. 실행 `kubectl describe pv` PV에 대한 세부 정보를 봅니다.

```
kubectl describe pv default-pvc-2-a8486

Name:            default-pvc-2-a8486
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: netapp.io/trident
                 volume.beta.kubernetes.io/storage-class: standard
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    standard
Status:          Bound
Claim:           default/pvc-2
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        1073741824
Node Affinity:   <none>
Message:
Source:
    Type:      NFS (an NFS mount that lasts the lifetime of a pod)
    Server:    10.xx.xx.xx
    Path:      /trid_1907_alpha_default_pvc_2_a8486
    ReadOnly:  false
```

PV는 을 사용하여 만들었습니다 `netapp.io/trident` 프로비저닝했으며 NFS 유형입니다. Astra Trident에서 제공하는 모든 새로운 기능을 지원하려면 이 PV를 CSI 유형으로 업그레이드해야 합니다.

3. 를 실행합니다 `tridentctl upgrade volume <name-of-trident-volume>` 레거시 Astra Trident 볼륨을 CSI 사양으로 업그레이드하는 명령입니다.

```
./tridentctl get volumes -n trident
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
|            NAME     |  SIZE   | STORAGE CLASS | PROTOCOL |
BACKEND UUID             | STATE  | MANAGED |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-3-a849e | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-1-a8475 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-4-a84de | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+

./tridentctl upgrade volume default-pvc-2-a8486 -n trident
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
|            NAME     |  SIZE   | STORAGE CLASS | PROTOCOL |
BACKEND UUID             | STATE  | MANAGED |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
```

4. A를 실행합니다 `kubectl describe pv` 볼륨이 CSI 볼륨인지 확인합니다.

```
kubectl describe pv default-pvc-2-a8486
Name:            default-pvc-2-a8486
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: csi.trident.netapp.io
                 volume.beta.kubernetes.io/storage-class: standard
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    standard
Status:          Bound
Claim:           default/pvc-2
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        1073741824
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume
source)
    Driver:            csi.trident.netapp.io
    VolumeHandle:      default-pvc-2-a8486
    ReadOnly:          false
    VolumeAttributes:      backendUUID=c5a6f6a4-b052-423b-80d4-
8fb491a14a22

internalName=trid_1907_alpha_default_pvc_2_a8486
                           name=default-pvc-2-a8486
                           protocol=file
Events:                <none>
```

Astra Trident를 제거합니다

Astra Trident의 설치 방식에 따라 여러 옵션을 사용하여 제거할 수 있습니다.

Helm을 사용하여 제거합니다

Helm을 사용하여 Astra Trident를 설치한 경우 를 사용하여 제거할 수 있습니다 `helm uninstall`.

```
#List the Helm release corresponding to the Astra Trident install.
helm ls -n trident
NAME          NAMESPACE       REVISION        UPDATED
STATUS          CHART                           APP VERSION
trident       trident         1               2021-04-20
00:26:42.417764794 +0000 UTC deployed        trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
helm uninstall trident -n trident
release "trident" uninstalled
```

Trident 연산자를 사용하여 제거합니다

운영자를 사용하여 Astra Trident를 설치한 경우 다음 중 하나를 수행하여 제거할 수 있습니다.

- 편집 **`TridentOrchestrator`** 제거 플래그를 설정하려면: 편집할 수 있습니다 `TridentOrchestrator` 그리고 설정합니다 `spec.uninstall=true`. 를 편집합니다 `TridentOrchestrator` 를 사용하여 를 설정합니다 `uninstall` 아래에 표시된 대로 플래그 지정:

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

  를 누릅니다 `uninstall` 플래그가 로 설정되어 있습니다 `true`Trident 운영자가 Trident를 제거하지만 Trident 자체 자체를 제거하지는 않습니다. 원하는 경우 트리엔오케스트레이터 를 청소하고 새 것을 만들어야 합니다 Trident를 다시 설치합니다.

- 삭제 **`TridentOrchestrator`**: 를 제거하여 `TridentOrchestrator` Astra Trident를 배포하는 데 사용된 CR은 작업자에게 Trident를 제거하도록 지시합니다. 작업자가 의 제거를 처리합니다 `TridentOrchestrator` 그런 다음 Astra Trident 구축과 디멘시작을 제거하고 설치의 일부로 생성한 Trident 포드를 삭제합니다. Astra Trident(CRD 생성 포함)를 완전히 제거하고 슬레이트 클린을 효과적으로 닦을 수 있습니다 `TridentOrchestrator` 을 전달합니다 `wipeout` 옵션을 선택합니다. 다음 예를 참조하십시오.

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

그러면 Astra Trident가 완전히 설치 제거되며, 백엔드 및 관리하는 볼륨과 관련된 모든 메타데이터가 지워집니다. 이후 설치는 새로 설치하는 것으로 처리됩니다.

> 전체 제거를 수행할 때에만 CRD를 지우는 것을 고려해야 합니다. 이 작업은 취소할 수 없습니다. 처음부터 새로 **Astra Trident** 설치를 생성하기 위해 사용하지 않는 한 **CRD**를 지우지 마십시오.

을 사용하여 를 제거합니다 `tridentctl`

를 실행합니다 `uninstall` 명령을 입력합니다 `tridentctl` 다음과 같이 CRD 및 관련 객체를 제외한 Astra Trident와 관련된 모든 리소스를 제거하므로 설치 프로그램을 다시 쉽게 실행하여 최신 버전으로 업데이트할 수 있습니다.

```
./tridentctl uninstall -n <namespace>
```

Astra Trident를 완전히 제거하려면 Astra Trident에서 생성한 CRD의 종료자를 제거하고 CRD를 삭제해야 합니다.

Astra Trident를 다운그레이드하십시오

Astra Trident의 이전 버전으로 다운그레이드하는 데 필요한 단계에 대해 알아보십시오.

다운그레이드 시점

다음과 같은 다양한 이유로 다운그레이드를 고려할 수 있습니다.

- 비상 계획
- 업그레이드로 인해 발견된 버그를 즉시 수정합니다
- 종속성 문제, 실패 및 불완전한 업그레이드

CRD를 사용하는 Astra Trident 릴리즈로 이전할 때는 다운그레이드를 고려해야 합니다. Astra Trident는 CRD를 사용하여 상태를 유지하기 때문에 생성된 모든 스토리지 요소(백엔드, 스토리지 클래스, PV 및 볼륨 스냅샷)는 에 기록된 데이터 대신 CRD 객체를 연결합니다 `trident` PV(이전 설치 버전의 Astra Trident에서 사용). 새로 생성된 PVS, 백엔드 및 스토리지 클래스는 모두 CRD 객체로 유지됩니다.

CRD를 사용하여 실행되는 Astra Trident 버전에 대해서만 다운그레이드를 시도하십시오(19.07 이상). 그러면 다운그레이드가 발생한 후 현재 Astra Trident 릴리스에 대해 수행된 작업이 표시됩니다.

다운그레이드를 하지 않는 경우

를 사용하는 Trident 릴리즈로 다운그레이드하면 안 됩니다 `etcd` 상태를 유지합니다(19.04 이하). 현재 Astra Trident 릴리즈를 통해 수행된 모든 작업은 다운그레이드 후 반영되지 않습니다. 새로 생성된 PVS는 이전 버전으로 되돌릴 때 사용할 수 없습니다. 이전 버전으로 돌아갈 때 Astra Trident에서 백엔드, PVS, 스토리지 클래스 및 볼륨 스냅샷(생성/업데이트/삭제)과 같은 객체에 대한 변경 사항을 볼 수 없습니다. 이전 버전으로 돌아가도 업그레이드되지 않은 경우 이전 릴리즈를 사용하여 이미 생성된 PVS에 대한 액세스가 중단되지 않습니다.

운영자를 통해 **Astra Trident**가 설치된 경우의 다운그레이드 프로세스

Trident 연산자를 사용하여 설치한 경우 다운그레이드 프로세스가 다르며 을 사용할 필요가 없습니다 `tridentctl`.

Trident 연산자를 사용하여 설치한 경우 Astra Trident를 다음 중 하나로 다운그레이드할 수 있습니다.

- 네임스페이스 범위 연산자를 사용하여 설치된 버전(20.07-20.10).
- 클러스터 범위 연산자(21.01 이상)를 사용하여 설치된 버전입니다.

클러스터 범위 연산자로 다운그레이드

Astra Trident를 클러스터 범위 운영자를 사용하는 릴리즈로 다운그레이드하려면 아래에 설명된 단계를 따르십시오.

단계

1. "Astra Trident를 제거합니다". 기존 설치를 완전히 제거하지 않는 한 **CRD**를 삭제하지 마십시오.
2. Trident 연산자는 사용 중인 Trident 버전에 연결된 연산자 매니페스트를 사용하여 삭제할 수 있습니다. 예를 들면, 다음과 같습니다. `https://github.com/NetApp/trident/tree/stable/vXX.XX /deploy/bundle.yaml` 위치 `vXX.XX` 은 버전 번호입니다(예 `v22.10`) 및 `bundle.yaml` 번들 YAML 파일 이름입니다.
3. 원하는 버전의 Astra Trident를 설치하여 다운그레이드를 계속합니다. 원하는 릴리스에 대한 설명서를 따릅니다.

네임스페이스 범위 연산자로 다운그레이드합니다

이 섹션에서는 네임스페이스 범위 연산자를 사용하여 설치되는 20.07 ~ 20.10 범위의 Astra Trident 릴리스로 다운그레이드하는 단계를 요약합니다.

단계

1. "Astra Trident를 제거합니다". 기존 설치를 완전히 제거하지 않는 한 **CRD**를 휘두리지 마십시오.
   를 확인합니다 `tridentorchestrator` 이(가) 삭제됩니다.

```
#Check to see if there are any tridentorchestrators present
kubectl get torc
NAME        AGE
trident     20h

#Looks like there is a tridentorchestrator that needs deleting
kubectl delete torc trident
tridentorchestrator.trident.netapp.io "trident" deleted
```

2. Trident 연산자는 사용 중인 Trident 버전에 연결된 연산자 매니페스트를 사용하여 삭제할 수 있습니다. 예를 들면, 다음과 같습니다. `https://github.com/NetApp/trident/tree/stable/vXX.XX /deploy/bundle.yaml` 위치 `vXX.XX` 은 버전 번호입니다(예 `v22.10`) 및 `bundle.yaml` 번들 YAML 파일 이름입니다.
3. 를 삭제합니다 `tridentorchestrator` CRD

```
#Check to see if ``tridentorchestrators.trident.netapp.io`` CRD is
present and delete it.

kubectl get crd tridentorchestrators.trident.netapp.io

NAME                                          CREATED AT
tridentorchestrators.trident.netapp.io   2021-01-21T21:11:37Z

kubectl delete crd tridentorchestrators.trident.netapp.io

customresourcedefinition.apiextensions.k8s.io
"tridentorchestrators.trident.netapp.io" deleted
```

Astra Trident가 제거되었습니다.

4. 원하는 버전을 설치하여 다운그레이드를 계속합니다. 원하는 릴리스에 대한 설명서를 따릅니다.

H제어 를 사용하여 다운그레이드합니다

다운그레이드하려면 을 사용합니다 `helm rollback` 명령. 다음 예를 참조하십시오.

```
helm rollback trident [revision #]
```

을 사용하여 **Astra Trident**를 설치한 경우의 다운그레이드 프로세스 `tridentctl`

을 사용하여 Astra Trident를 설치한 경우 `tridentctl`다운그레이드 프로세스는 다음 단계를 포함합니다. 이 시퀀스는 Astra Trident 21.07에서 20.07로 이동하는 다운그레이드 프로세스를 안내합니다.

다운그레이드를 시작하기 전에 Kubernetes 클러스터의 스냅샷을 만들어야 합니다 `etcd`. 이를 통해 Astra Trident의 CRD의 현재 상태를 백업할 수 있습니다.

단계

1. 를 사용하여 Trident가 설치되었는지 확인합니다 `tridentctl`. Astra Trident의 설치 방법을 잘 모르는 경우 다음 간단한 테스트를 실행하십시오.
   a. Trident 네임스페이스에 있는 포드를 나열합니다.
   b. 클러스터에서 실행 중인 Astra Trident의 버전을 확인합니다. 를 사용할 수 있습니다 `tridentctl` 또는 Trident Pod에 사용된 이미지를 살펴보십시오.
   c. 가 표시되지 않는 경우 * a `tridentOrchestrator`, (또는) a `tridentprovisioner`, (또는) 이름이 인 포드 `trident-operator-xxxxxxxxxx-xxxxx`, Astra Trident * 가 와 함께 설치됩니다 `tridentctl`.
2. 기존 인프라와 함께 Astra Trident를 제거합니다 `tridentctl` 바이너리. 이 경우 21.07 바이너리로 를 제거합니다.

```
tridentctl version -n trident
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.0        | 21.07.0        |
+----------------+----------------+

tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted Trident daemonset.
INFO Deleted Trident service.
INFO Deleted Trident secret.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Deleted pod security policy.
podSecurityPolicy=tridentpods
INFO The uninstaller did not delete Trident's namespace in case it is
going to be reused.
INFO Trident uninstallation succeeded.
```

3. 이 작업이 완료되면 원하는 버전의 Trident 바이너리(이 예: 20.07)를 얻고 이를 사용하여 Astra Trident를 설치합니다. 에 대한 사용자 지정 YAML을 생성할 수 있습니다 "맞춤형 설치" 필요한 경우

```
cd 20.07/trident-installer/
./tridentctl install -n trident-ns
INFO Created installer service account.
serviceaccount=trident-installer
INFO Created installer cluster role.               clusterrole=trident-
installer
INFO Created installer cluster role binding.
clusterrolebinding=trident-installer
INFO Created installer configmap.                  configmap=trident-
installer
...
...
INFO Deleted installer cluster role binding.
INFO Deleted installer cluster role.
INFO Deleted installer service account.
```

다운그레이드 프로세스가 완료되었습니다.

저작권 정보

Copyright © 2025 NetApp, Inc. All Rights Reserved. 미국에서 인쇄됨 본 문서의 어떠한 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 수단(복사, 녹음, 녹화 또는 전자 검색 시스템에 저장하는 것을 비롯한 그래픽, 전자적 또는 기계적 방법)으로도 복제될 수 없습니다.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이센스와 고지사항이 적용됩니다.

본 소프트웨어는 NetApp에 의해 '있는 그대로' 제공되며 상품성 및 특정 목적에의 적합성에 대한 명시적 또는 묵시적 보증을 포함하여(이에 제한되지 않음) 어떠한 보증도 하지 않습니다. NetApp은 대체품 또는 대체 서비스의 조달, 사용 불능, 데이터 손실, 이익 손실, 영업 중단을 포함하여(이에 국한되지 않음), 이 소프트웨어의 사용으로 인해 발생하는 모든 직접 및 간접 손해, 우발적 손해, 특별 손해, 징벌적 손해, 결과적 손해의 발생에 대하여 그 발생 이유, 책임론, 계약 여부, 엄격한 책임, 불법 행위(과실 또는 그렇지 않은 경우)와 관계없이 어떠한 책임도 지지 않으며, 이와 같은 손실의 발생 가능성이 통지되었다 하더라도 마찬가지입니다.

NetApp은 본 문서에 설명된 제품을 언제든지 예고 없이 변경할 권리를 보유합니다. NetApp은 NetApp의 명시적인 서면 동의를 받은 경우를 제외하고 본 문서에 설명된 제품을 사용하여 발생하는 어떠한 문제에도 책임을 지지 않습니다. 본 제품의 사용 또는 구매의 경우 NetApp에서는 어떠한 특허권, 상표권 또는 기타 지적 재산권이 적용되는 라이센스도 제공하지 않습니다.

본 설명서에 설명된 제품은 하나 이상의 미국 특허, 해외 특허 또는 출원 중인 특허로 보호됩니다.

제한적 권리 표시: 정부에 의한 사용, 복제 또는 공개에는 DFARS 252.227-7013(2014년 2월) 및 FAR 52.227-19(2007년 12월)의 기술 데이터-비상업적 품목에 대한 권리(Rights in Technical Data -Noncommercial Items) 조항의 하위 조항 (b)(3)에 설명된 제한사항이 적용됩니다.

여기에 포함된 데이터는 상업용 제품 및/또는 상업용 서비스(FAR 2.101에 정의)에 해당하며 NetApp, Inc.의 독점 자산입니다. 본 계약에 따라 제공되는 모든 NetApp 기술 데이터 및 컴퓨터 소프트웨어는 본질적으로 상업용이며 개인 비용만으로 개발되었습니다. 미국 정부는 데이터가 제공된 미국 계약과 관련하여 해당 계약을 지원하는 데에만 데이터에 대한 전 세계적으로 비독점적이고 양도할 수 없으며 재사용이 불가능하며 취소 불가능한 라이센스를 제한적으로 가집니다. 여기에 제공된 경우를 제외하고 NetApp, Inc.의 사전 서면 승인 없이는 이 데이터를 사용, 공개, 재생산, 수정, 수행 또는 표시할 수 없습니다. 미국 국방부에 대한 정부 라이센스는 DFARS 조항 252.227-7015(b)(2014년 2월)에 명시된 권한으로 제한됩니다.

상표 정보

NETAPP, NETAPP 로고 및 http://www.netapp.com/TM에 나열된 마크는 NetApp, Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.