Astra Trident를 관리 및 모니터링하십시오

Astra Trident

NetApp

April 03, 2024

목차

Astra Trident를 관리 및 모니터링하십시오

Astra Trident를 업그레이드합니다

Astra Trident를 업그레이드합니다

Astra Trident는 분기별 릴리스 케이던스를 따르며, 매년 4개의 주요 릴리즈를 제공합니다. 각각의 새 릴리스는 이전 릴리스에서 빌드되며 새로운 기능, 성능 향상, 버그 수정 및 개선 사항을 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.
- Astra Trident 23.07 이상에서는 v1 볼륨 스냅샷이 필요하며 알파 또는 베타 스냅샷을 더 이상 지원하지 않습니다.
- 에서 Google Cloud용 Cloud Volumes Service를 생성한 경우 "CVS 서비스 유형입니다"을 사용하려면 백엔드 구성을 업데이트해야 합니다 `standardsw` 또는 `zoneredundantstandardsw` Astra Trident 23.01에서 업그레이드할 때의 서비스 수준입니다. 를 업데이트하지 못했습니다 `serviceLevel` 백엔드에서 볼륨이 실패할 수 있습니다. 을 참조하십시오 "CVS 서비스 유형 샘플" 를 참조하십시오.
- 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 22.07(또는 22.07 도트 릴리스)에서 23.07로 직접 업그레이드할 수 있습니다.
- 릴리스 4개가 아닌 다른 릴리스에서 업그레이드하는 경우 여러 단계로 업그레이드를 수행합니다. 의 업그레이드 지침을 사용합니다 "이전 버전" 에서 4개의 릴리즈 윈도우에 맞는 최신 릴리즈로 업그레이드하는 중입니다. 예를 들어, 21.07을 실행 중이고 23.07로 업그레이드하려는 경우:
  a. 먼저 21.07에서 22.07로 업그레이드합니다.
  b. 그런 다음 22.07에서 23.07로 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

처음에 Astra Trident를 설치하는 데 사용한 버전을 확인하려면:

1. 사용 `kubectl get pods -n 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

일반적으로 초기 설치에 사용한 것과 동일한 방법으로 업그레이드해야 하지만 "설치 방법 간에 이동합니다". Astra Trident를 업그레이드할 수 있는 두 가지 옵션이 있습니다.

- "Trident 연산자를 사용하여 업그레이드합니다"

검토할 것을 제안합니다 "운영자 업그레이드 워크플로우를 이해합니다" 작업자와 업그레이드하기 전에

*

운영자와 함께 업그레이드하십시오

운영자 업그레이드 워크플로우를 이해합니다

Trident 연산자를 사용하여 Astra Trident를 업그레이드하기 전에 업그레이드 중에 발생하는 백그라운드 프로세스를 이해해야 합니다. 여기에는 Trident 컨트롤러, 컨트롤러 Pod 및 노드 Pod, 롤링 업데이트를 사용하는 노드 DemonSet의 변경 사항이 포함됩니다.

Trident 운영자 업그레이드 처리

많은 것 중 하나입니다 "Trident 연산자를 사용할 때의 이점" Astra Trident를 설치 및 업그레이드하는 데 사용되는 것은 기존의 마운트된 볼륨을 중단하지 않고 Astra Trident 및 Kubernetes 개체를 자동으로 처리하는 것입니다. 이를 통해 Astra Trident는 다운타임 없이 업그레이드를 지원할 수 있으며 또는 "_롤링 업데이트 _". 특히, Trident 운영자는 Kubernetes 클러스터와 통신하여 다음을 수행합니다.

- Trident 컨트롤러 배포 및 노드 DemonSet을 삭제하고 다시 만듭니다.
- Trident 컨트롤러 Pod 및 Trident 노드 Pod를 새로운 버전으로 교체합니다.

- 노드가 업데이트되지 않으면 나머지 노드가 업데이트됩니다.
- 실행 중인 Trident 노드 Pod가 있는 노드에서만 볼륨을 마운트할 수 있습니다.

Kubernetes 클러스터의 Astra Trident 아키텍처에 대한 자세한 내용은 을 참조하십시오 "Astra Trident 아키텍처".

운영자 업그레이드 워크플로우

Trident 연산자를 사용하여 업그레이드를 시작하는 경우:

1. Trident 운영자 *:
   a. 현재 설치된 Astra Trident 버전(version_n_)을 감지합니다.
   b. CRD, RBAC, Trident SVC를 포함한 모든 Kubernetes 오브젝트를 업데이트합니다.
   c. 버전_n_에 대한 Trident 컨트롤러 배포를 삭제합니다.
   d. 버전_n+1_에 대한 Trident 컨트롤러 배포를 생성합니다.
2. * Kubernetes * 는 _n+1_용 Trident 컨트롤러 포드를 생성합니다.
3. Trident 운영자 *:
   a. _n_에 대한 Trident 노드 데모 세트를 삭제합니다. 운용자는 Node Pod 종료를 기다리지 않는다.
   b. _n+1_에 대한 Trident 노드 데모 세트를 생성합니다.
4. * Kubernetes * 는 Trident 노드 Pod_n_을(를) 실행하지 않는 노드에서 Trident 노드 Pod를 생성합니다. 따라서 노드에 여러 버전의 Trident 노드 Pod가 둘 이상 있지 않도록 합니다.

Trident 운영자 설치를 업그레이드합니다

수동으로 또는 Helm을 사용하여 Trident 연산자를 사용하여 Astra Trident를 업그레이드할 수 있습니다. Trident 운영자 설치에서 다른 Trident 운영자 설치로 업그레이드하거나, 에서 업그레이드할 수 있습니다 `tridentctl` Trident 운영자 버전에 설치. 검토 "업그레이드 방법을 선택합니다" Trident 운영자 설치를 업그레이드하기 전에.

수동 설치를 업그레이드합니다

클러스터 범위 Trident 운영자 설치에서 다른 클러스터 범위 Trident 운영자 설치로 업그레이드할 수 있습니다. Astra Trident 버전 21.01 이상에서는 클러스터 범위 연산자를 사용합니다.

네임스페이스 범위 운영자(버전 20.07~20.10)를 사용하여 설치된 Astra Trident에서 업그레이드하려면 의 업그레이드 지침을 사용하십시오 "설치된 버전" 기술입니다.

이 작업에 대해

Trident는 운영자를 설치하고 Kubernetes 버전에 대한 관련 오브젝트를 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".

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 23.04에서 업그레이드하는 경우 다음 명령을 실행합니다.

```
kubectl delete -f 23.04/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 환경에 적합한 번들 YAML 파일을 사용하여 Astra Trident를 설치합니다. 여기서 _<bundle.yaml>_is는 `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml` Kubernetes 버전을 기반으로 합니다. 예를 들어 Astra Trident 23.07을 설치하는 경우 다음 명령을 실행합니다.

```
kubectl create -f 23.07.1/trident-installer/deploy/<bundle.yaml> -n
trident
```

Helm 설치를 업그레이드합니다

Astra Trident Helm 설치를 업그레이드할 수 있습니다.

> 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-23.07.1.tgz` 업그레이드하려는 버전을 반영합니다.

```
helm upgrade <name> trident-operator-23.07.1.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-23.07.1-custom.tgz --set
tridentDebug=true
```

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

에서 업그레이드 `tridentctl` **Trident** 운영자에 설치

에서 Trident 운영자의 최신 릴리즈로 업그레이드할 수 있습니다 `tridentctl` 설치: 기존 백엔드 및 PVC를 자동으로 사용할 수 있습니다.

설치 방법 간에 전환하기 전에 를 참조하십시오 "설치 방법 간 이동"

단계

1. 최신 Astra Trident 릴리스를 다운로드하십시오.

```
# Download the release required [23.07.1]
mkdir 23.07.1
cd 23.07.1
wget
https://github.com/NetApp/trident/releases/download/v22.01.1/trident-
installer-23.07.1.tar.gz
tar -xf trident-installer-23.07.1.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-name.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:                v23.07.1
```

tridentctl로 업그레이드하십시오

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

이 작업에 대해

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.

Astra Trident를 모니터링합니다

Astra Trident는 Astra Trident 성능을 모니터링하는 데 사용할 수 있는 Prometheus 메트릭 엔드포인트 세트를 제공합니다.

개요

Astra Trident에서 제공하는 메트릭을 통해 다음을 수행할 수 있습니다.

- Astra Trident의 상태 및 구성을 계속 확인하십시오. 성공적인 작업이 어떻게 이루어지는지, 예상대로 백엔드와 통신할 수 있는지 확인할 수 있습니다.
- 백엔드 사용 정보를 검토하고 백엔드에서 프로비저닝되는 볼륨 수와 사용된 공간 등을 파악합니다.
- 사용 가능한 백엔드에 프로비저닝된 볼륨 양의 매핑을 유지합니다.
- 성과 추적. Astra Trident가 백엔드 및 작업을 수행하는 데 걸리는 시간을 확인할 수 있습니다.

기본적으로 Trident의 메트릭은 타겟 포트에 표시됩니다 `8001` 를 누릅니다 `/metrics` 엔드포인트. Trident가 설치된 경우 이러한 메트릭은 기본적으로 * 활성화됩니다.

필요한 것

- Astra Trident가 설치된 Kubernetes 클러스터
- 프로메테우스(Prometheus) 인스턴스. 이것은 일 수 있습니다 "컨테이너형 Prometheus 구축" 또는 Prometheus를

로 실행하도록 선택할 수 있습니다 "네이티브 애플리케이션".

1단계: Prometheus 목표를 정의합니다

메트릭을 수집하고 백엔드 Astra Trident가 관리하는, 생성하는 볼륨 등에 대한 정보를 얻으려면 Prometheus 타겟을 정의해야 합니다. 여기 "블로그" Prometheus 및 Grafana를 Astra Trident와 함께 사용하여 메트릭을 검색하는 방법에 대해 설명합니다. 블로그에서 Kubernetes 클러스터의 운영자로서 Prometheus를 실행하고 ServiceMonitor를 생성하여 Astra Trident 메트릭을 얻는 방법을 설명합니다.

2단계: Prometheus ServiceMonitor를 만듭니다

Trident 메트릭을 소모하려면 을 감시하는 Prometheus ServiceMonitor를 만들어야 합니다 `trident-csi` 에 대한 서비스와 수신 `metrics` 포트. 샘플 ServiceMonitor의 모양은 다음과 같습니다.

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
      release: prom-operator
  spec:
    jobLabel: trident
    selector:
      matchLabels:
        app: controller.csi.trident.netapp.io
    namespaceSelector:
      matchNames:
      - trident
    endpoints:
    - port: metrics
      interval: 15s
```

이 ServiceMonitor 정의는 에서 반환된 메트릭을 검색합니다 `trident-csi` 을(를) 위해 특별히 찾습니다 `metrics` 서비스의 끝점입니다. 따라서 이제 Prometheus가 Astra Trident를 이해하도록 구성되었습니다
메트릭:

Astra Trident에서 직접 제공하는 메트릭 외에도 kubelet은 많은 정보를 제공합니다 `kubelet_volume_*` 자체 메트릭 엔드포인트를 통해 측정 Kubelet는 연결된 볼륨, Pod 및 처리하는 기타 내부 작업에 대한 정보를 제공할 수 있습니다. 을 참조하십시오 "여기".

3단계: PromQL을 사용하여 Trident 메트릭 쿼리

PromQL은 시계열 또는 표 형식 데이터를 반환하는 식을 만드는 데 적합합니다.

다음은 사용할 수 있는 몇 가지 PromQL 쿼리입니다.

Trident 상태 정보를 가져옵니다

- Astra Trident** 의 HTTP 2XX 응답률

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- Astra Trident에서 상태 코드를 통해 얻은 REST 응답의 비율**

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Astra Trident** 에 의해 수행된 작업의 평균 지속 시간(ms)

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Astra Trident 사용 정보를 확인하십시오

- 평균 볼륨 크기

```
trident_volume_allocated_bytes/trident_volume_count
```

- 각 백엔드에서 프로비저닝된 총 볼륨 공간

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

개별 볼륨 사용량을 가져옵니다

> 이 기능은 kubelet 메트릭도 수집한 경우에만 사용할 수 있습니다.

- 각 볼륨에 사용된 공간의 비율

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Astra Trident AutoSupport 텔레메트리 에 대해 자세히 알아보십시오

기본적으로 Astra Trident는 Prometheus 메트릭 및 기본 백엔드 정보를 매일 NetApp에 보냅니다.

- Astra Trident가 Prometheus 메트릭 및 기본 백엔드 정보를 NetApp에 보내는 것을 중지하려면 을 전달합니다 `--silence-autosupport` Astra Trident를 설치하는 동안 플래그를 지정합니다.
- Astra Trident는 또한 을 통해 컨테이너 로그를 NetApp Support On-Demand로 보낼 수 있습니다 `tridentctl send autosupport`. 로그를 업로드하려면 Astra Trident를 트리거해야 합니다. 로그를 제출하기 전에 NetApp의 내용에 동의해야 합니다
  "개인 정보 보호 정책".
- 지정되지 않은 경우 Astra Trident는 지난 24시간 동안 로그를 가져옵니다.
- 을 사용하여 로그 보존 기간을 지정할 수 있습니다 `--since` 깃발. 예를 들면 다음과 같습니다. `tridentctl send autosupport --since=1h`. 이 정보는 를 통해 수집 및 전송됩니다 `trident-autosupport` 컨테이너
  Astra Trident와 함께 설치됩니다. 에서 컨테이너 이미지를 얻을 수 있습니다 "Trident AutoSupport를 누릅니다".
- Trident AutoSupport는 개인 식별 정보(PII) 또는 개인 정보를 수집하거나 전송하지 않습니다. 이 제품은 와 함께 제공됩니다 "EULA" Trident 컨테이너 이미지 자체에는 적용되지 않습니다. 데이터 보안 및 신뢰에 대한 NetApp의 노력에 대해 자세히 알아볼 수 있습니다 "여기".

Astra Trident에서 보낸 페이로드의 예는 다음과 같습니다.

```
---
items:
- backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
  protocol: file
  config:
    version: 1
    storageDriverName: ontap-nas
    debug: false
    debugTraceFlags:
    disableDelete: false
    serialNumbers:
    - nwkvzfanek_SN
    limitVolumeSize: ''
  state: online
  online: true
```

- AutoSupport 메시지는 NetApp의 AutoSupport 엔드포인트로 전송됩니다. 개인 레지스트리를 사용하여 컨테이너 이미지를 저장하는 경우 를 사용할 수 있습니다 `--image-registry` 깃발.
- 또한 설치 YAML 파일을 생성하여 프록시 URL을 구성할 수도 있습니다. 이 작업은 를 사용하여 수행할 수 있습니다 `tridentctl install --generate-custom-yaml` YAML 파일을 생성하고 을 추가합니다 `--proxy-url` 에 대한 인수 `trident-autosupport` 컨테이너 인합니다 `trident-deployment.yaml`.

Astra Trident 메트릭을 비활성화합니다

- 메트릭을 보고하지 않으려면 을 사용하여 사용자 지정 YAML을 생성해야 합니다 `--generate-custom-yaml` Flag)을 사용하여 를 제거합니다 `--metrics` 에 대해 호출되는 플래그 `trident-main` 컨테이너.

Astra Trident를 제거합니다

Astra Trident를 설치하는 데 사용한 Astra Trident를 제거하는 데 동일한 방법을 사용해야 합니다.

이 작업에 대해

- 업그레이드, 종속성 문제 또는 실패 또는 불완전한 업그레이드 후에 관찰된 버그에 대한 수정이 필요한 경우 Astra Trident를 제거하고 해당 지침에 따라 이전 버전을 다시 설치해야 합니다 "버전". 이전 버전으로 _download_하는 유일한 권장 방법입니다.
- 간편한 업그레이드 및 재설치를 위해 Astra Trident를 제거해도 Astra Trident에서 생성된 CRD 또는 관련 개체는 제거되지 않습니다. Astra Trident 및 모든 데이터를 완전히 제거해야 하는 경우 를 참조하십시오 "Astra Trident 및 CRD를 완전히 제거합니다".

시작하기 전에

Kubernetes 클러스터를 사용 중단하는 경우, 를 제거하기 전에 Astra Trident에서 생성된 볼륨을 사용하는 모든 애플리케이션을 삭제해야 합니다. 따라서 PVC가 삭제되기 전에 Kubernetes 노드에 게시되지 않습니다.

원래 설치 방법을 확인합니다

Astra Trident를 설치할 때 사용한 것과 동일한 방법을 사용하여 제거해야 합니다. 제거하기 전에 Astra Trident를 원래 설치할 때 사용한 버전을 확인하십시오.

1. 사용 `kubectl get pods -n 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.

Trident 운영자 설치를 제거합니다

수동 또는 Helm을 사용하여 트라이덴트 작업자 설치를 제거할 수 있습니다.

수동 설치를 제거합니다

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

1. 편집 **`TridentOrchestrator` CR** 및 제거 플래그 설정:

```
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. Trident를 다시 설치하려면 해당 Trident를 정리하고 새 AgentOrchestrator를 생성해야 합니다.

2. 삭제 **`TridentOrchestrator`**: 를 제거합니다 `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.

```
kubectl delete -f deploy/<bundle.yaml> -n <namespace>
```

Helm 설치를 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
```

를 제거합니다 `tridentctl` 설치

를 사용합니다 `uninstall` 명령을 입력합니다 `tridentctl` CRD 및 관련 오브젝트를 제외하고 Astra Trident에 연결된 모든 리소스를 제거하려면 다음과 같이 하십시오.

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