Trident 관리 및 모니터링

Trident

NetApp
July 01, 2026

목차

Trident 관리 및 모니터링

Trident 업그레이드

Trident 업그레이드

24.02 릴리스부터 Trident는 4개월 주기로 릴리스를 진행하며, 매년 세 번의 주요 릴리스를 제공합니다. 각 새 릴리스는 이전 릴리스를 기반으로 새로운 기능, 성능 향상, 버그 수정 및 개선 사항을 제공합니다. Trident의 새로운 기능을 활용하려면 1년에 한 번 이상 업그레이드하는 것이 좋습니다.

업그레이드 전 고려 사항

Trident의 최신 릴리스로 업그레이드할 때 다음 사항을 고려하십시오.

- 주어진 Kubernetes 클러스터 내의 모든 네임스페이스에는 하나의 Trident 인스턴스만 설치되어야 합니다.
- Trident 23.07 이상에서는 v1 볼륨 스냅샷이 필요하며 알파 또는 베타 스냅샷은 더 이상 지원되지 않습니다.
- 업그레이드할 때, Trident에서 사용하는 `parameter.fsType`에 `StorageClasses`를 제공하는 것이 중요합니다. 기존 볼륨에 영향을 주지 않고 `StorageClasses`를 삭제하고 다시 생성할 수 있습니다.
  - 이는 SAN 볼륨에 대한 "보안 컨텍스트" 적용을 위한 필수 요건입니다.
  - 샘플 입력 디렉토리에는 `storage-class-basic.yaml.templ` 및 `storage-class-bronze-default.yaml`와 같은 예제가 포함되어 있습니다.
  - 자세한 내용은 "알려진 문제"을 참조하십시오.

1단계**:** 버전 선택

Trident 버전은 날짜 기반 `YY.MM` 명명 규칙을 따르며, 여기서 "YY"는 연도의 마지막 두 자리 숫자이고 "MM"은 월을 나타냅니다. 도트 릴리스는 `YY.MM.X` 규칙을 따르며, 여기서 "X"는 패치 레벨을 나타냅니다. 업그레이드할 버전은 업그레이드하려는 현재 버전을 기준으로 선택합니다.

- 설치된 버전에서 4개 릴리스 이내의 대상 릴리스로 직접 업그레이드할 수 있습니다. 예를 들어 24.06(또는 24.06의 모든 마이너 버전)에서 25.06으로 직접 업그레이드할 수 있습니다.
- 4개 릴리스 기간 외의 버전에서 업그레이드하는 경우 여러 단계를 거쳐 업그레이드하십시오. "이전 버전"에서 업그레이드하는 경우 해당 버전의 업그레이드 지침을 사용하여 4개 릴리스 기간에 맞는 최신 릴리스로 업그레이드하십시오. 예를 들어 23.07 버전을 실행 중이고 25.06으로 업그레이드하려는 경우:
  a. 먼저 23.07에서 24.06으로 업그레이드하십시오.
  b. 그다음 24.06에서 25.06으로 업그레이드하세요.

> OpenShift Container Platform에서 Trident 오퍼레이터를 사용하여 업그레이드할 경우 Trident 21.01.1 이상으로 업그레이드해야 합니다. 21.01.0과 함께 릴리스된 Trident 오퍼레이터에는 알려진 문제가 있으며 이 문제는 21.01.1에서 수정되었습니다. 자세한 내용은 "GitHub의 문제 세부 정보"을 참조하십시오.

2단계**:** 원래 설치 방법을 확인합니다

Trident를 원래 설치하는 데 사용한 버전을 확인하려면:

1. `kubectl get pods -n trident`를 사용하여 Pod를 검사합니다.
   - 오퍼레이터 포드가 없는 경우 Trident가 `tridentctl`을 사용하여 설치되었습니다.
   - 오퍼레이터 포드가 있는 경우, Trident는 수동으로 또는 Helm을 사용하여 Trident 오퍼레이터를 통해 설치되었습니다.
2. 운영자 포드가 있는 경우 `kubectl describe torc`를 사용하여 Helm을 통해 Trident가 설치되었는지 확인합니다.
   - Helm 레이블이 있는 경우 Trident는 Helm을 사용하여 설치되었습니다.
   - Helm 레이블이 없으면 Trident 운영자를 사용하여 Trident를 수동으로 설치한 것입니다.

3단계**:** 업그레이드 방법 선택

일반적으로 초기 설치 시 사용했던 방법과 동일한 방법으로 업그레이드하는 것이 좋지만, "설치 방법 간 이동". Trident를 업그레이드하는 두 가지 옵션이 있습니다.

- "Trident 운영자를 사용하여 업그레이드하세요"

> 운영자와 함께 업그레이드하기 전에 "운영자 업그레이드 워크플로 이해"을(를) 검토하는 것이 좋습니다.

*

운영자를 통해 업그레이드

운영자 업그레이드 워크플로 이해

Trident 오퍼레이터를 사용하여 Trident를 업그레이드하기 전에 업그레이드 중에 발생하는 백그라운드 프로세스를 이해해야 합니다. 여기에는 롤링 업데이트를 가능하게 하는 Trident 컨트롤러, 컨트롤러 Pod 및 노드 Pod, 노드 DaemonSet의 변경 사항이 포함됩니다.

Trident 운영자 업그레이드 처리

Trident를 설치 및 업그레이드하는 많은 "Trident 운영자 사용의 이점" 중 하나는 기존에 마운트된 볼륨에 영향을 주지 않고 Trident 및 Kubernetes 객체를 자동으로 처리한다는 점입니다. 이러한 방식으로 Trident는 다운타임 없이 업그레이드를 지원할 수 있습니다("롤링 업데이트"). 특히, Trident 오퍼레이터는 다음을 위해 Kubernetes 클러스터와 통신합니다.

- Trident Controller 배포 및 노드 DaemonSet을 삭제하고 다시 생성합니다.
- Trident Controller Pod와 Trident Node Pod를 새 버전으로 교체하십시오.
  - 노드가 업데이트되지 않더라도 나머지 노드의 업데이트가 차단되지 않습니다.
  - 실행 중인 Trident Node Pod가 있는 노드만 볼륨을 마운트할 수 있습니다.

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

Operator 업그레이드 워크플로

Trident 운영자를 사용하여 업그레이드를 시작할 때:

1. **Trident** 운영자:
   a. 현재 설치된 Trident 버전(*n*)을 감지합니다.
   b. CRD, RBAC 및 Trident SVC를 포함한 모든 Kubernetes 객체를 업데이트합니다.
   c. 버전 _n_에 대한 Trident Controller 배포를 삭제합니다.
   d. 버전 _n+1_에 대한 Trident Controller 배포를 생성합니다.
2. *Kubernetes*는 _n+1_에 대한 Trident Controller Pod를 생성합니다.
3. **Trident** 운영자:
   a. _n_에 대한 Trident 노드 DaemonSet을 삭제합니다. 운영자는 노드 Pod 종료를 기다리지 않습니다.
   b. _n+1_에 대한 Trident 노드 데몬셋을 생성합니다.
4. *Kubernetes*는 Trident Node Pod _n_이 실행 중이지 않은 노드에 Trident Node Pod을 생성합니다. 이를 통해 노드에는 어떤 버전이든 Trident Node Pod이 두 개 이상 존재하지 않게 됩니다.

Trident Operator 또는 Helm을 사용하여 Trident 설치를 업그레이드하세요

Trident 운영자를 사용하여 수동으로 또는 Helm을 통해 Trident를 업그레이드할 수 있습니다. Trident 운영자 설치에서 다른 Trident 운영자 설치로 업그레이드하거나 `tridentctl` 설치에서 Trident 운영자 버전으로 업그레이드할 수 있습니다. Trident 운영자 설치를 업그레이드하기 전에 "업그레이드 방법을 선택하세요"를 검토하십시오.

수동 설치 업그레이드

클러스터 범위 Trident 운영자 설치에서 다른 클러스터 범위 Trident 운영자 설치로 업그레이드할 수 있습니다. 모든 Trident 버전은 클러스터 범위 운영자를 사용합니다.

> ⓘ 네임스페이스 범위 연산자를 사용하여 설치된 Trident(버전 20.07~20.10)에서 업그레이드하려면 "설치된 버전" Trident의 업그레이드 지침을 사용하십시오.

이 작업 정보

Trident는 Kubernetes 버전에 맞는 오퍼레이터를 설치하고 관련 객체를 생성하는 데 사용할 수 있는 번들 파일을 제공합니다.

- Kubernetes 1.25 이상 버전을 실행하는 클러스터의 경우 "bundle_post_1_25.yaml"를 사용하십시오.

시작하기 전에

"지원되는 Kubernetes 버전"을(를) 실행하는 Kubernetes 클러스터를 사용하고 있는지 확인하십시오.

단계

1. Trident 버전을 확인하세요:

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

2. `operator.yaml`, `tridentorchestrator_cr.yaml`, `post_1_25_bundle.yaml`를 업그레이드할 버전(예: 25.06)에 맞는 레지스트리 및 이미지 경로와 올바른 시크릿으로 업데이트하십시오.
3. 현재 Trident 인스턴스를 설치하는 데 사용된 Trident 오퍼레이터를 삭제하십시오. 예를 들어 25.02에서 업그레이드하는 경우 다음 명령을 실행하십시오.

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. `TridentOrchestrator` 속성을 사용하여 초기 설치를 사용자 지정한 경우 `TridentOrchestrator` 객체를 편집하여 설치 매개 변수를 수정할 수 있습니다. 여기에는 오프라인 모드용 미러링된 Trident 및 CSI 이미지 레지스트리 지정, 디버그 로그 활성화 또는 이미지 풀 시크릿 지정을 위한 변경 사항이 포함될 수 있습니다.
5. Kubernetes 버전에 따라 _<bundle.yaml>_가 `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml`인 환경에 맞는 번들 YAML 파일을 사용하여 Trident를 설치합니다. 예를 들어 Trident 25.06.0을 설치하는 경우 다음 명령을 실행합니다.

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. Trident torc를 수정하여 이미지 25.06.0을 포함시키세요.

Helm 설치 업그레이드

Trident Helm 설치를 업그레이드할 수 있습니다.

> Trident가 설치된 Kubernetes 클러스터를 1.24에서 1.25 이상 버전으로 업그레이드할 때는 클러스터를 업그레이드하기 전에 values.yaml을 업데이트하여 `excludePodSecurityPolicy`을 `true`로 설정하거나 `--set excludePodSecurityPolicy=true`을 `helm upgrade` 명령에 추가해야 합니다.

Kubernetes 클러스터를 1.24에서 1.25로 업그레이드했지만 Trident helm을 업그레이드하지 않은 경우 helm 업그레이드가 실패합니다. helm 업그레이드를 성공적으로 완료하려면 다음 단계를 사전 수행해야 합니다.

1. https://github.com/helm/helm-mapkubeapis에서 helm-mapkubeapis 플러그인을 설치합니다.
2. Trident가 설치된 네임스페이스에서 Trident 릴리스에 대한 사전 실행(dry run)을 수행합니다. 이렇게 하면 정리될 리소스 목록이 표시됩니다.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. 정리 작업을 위해 helm을 사용하여 전체 실행을 수행하십시오.

```
helm mapkubeapis trident  --namespace trident
```

단계

1. "Helm을 사용하여 Trident를 설치했습니다"한 경우 `helm upgrade trident netapp-trident/trident-operator --version 100.2602.0`를 사용하여 한 단계로 업그레이드할 수 있습니다. Helm 리포지토리를 추가하지 않았거나 업그레이드에 사용할 수 없는 경우:

   a. 최신 Trident 릴리스를 "GitHub의 *Assets* 섹션"에서 다운로드하십시오.

   b. `helm upgrade` 명령을 사용합니다. 여기서 `trident-operator-26.02.0.tgz`는 업그레이드할 버전을 나타냅니다.

   ```
   helm upgrade <name> trident-operator-26.02.0.tgz
   ```

   초기 설치 중에 사용자 지정 옵션(예: Trident 및 CSI 이미지에 대한 개인 미러링 레지스트리 지정)을 설정한 경우 `helm upgrade` 명령을 `--set`를 사용하여 추가하여 해당 옵션이 업그레이드 명령에 포함되도록 해야 합니다. 그렇지 않으면 값이 기본값으로 재설정됩니다.

2. `helm list`을 실행하여 차트와 앱 버전이 모두 업그레이드되었는지 확인합니다. `tridentctl logs`를 실행하여 디버그 메시지를 검토합니다.

`tridentctl` 설치에서 **Trident operator**로 업그레이드

`tridentctl` 설치에서 `Trident` 운영자의 최신 릴리스로 업그레이드할 수 있습니다. 기존 백엔드와 `PVC`는 자동으로 사용할 수 있습니다.

설치 방법을 전환하기 전에 "설치 방법 간 이동"을(를) 검토하십시오.

단계

1. 최신 Trident 릴리스를 다운로드하세요.

   ```
   # Download the release required [26.02.0]
   mkdir 26.02.0
   cd 26.02.0
   wget
   https://github.com/NetApp/trident/releases/download/v26.02.0/trident-
   installer-26.02.0.tar.gz
   tar -xf trident-installer-26.02.0.tar.gz
   cd trident-installer
   ```

2. 매니페스트에서 `tridentorchestrator` CRD를 생성합니다.

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

3. 클러스터 범위 오퍼레이터를 동일한 네임스페이스에 배포합니다.

```
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. Trident 설치를 위한 `TridentOrchestrator` CR을 생성합니다.

```
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 확인하십시오.

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

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v26.02.0
```

tridentctl로 업그레이드합니다

`tridentctl`를 사용하여 기존 Trident 설치를 쉽게 업그레이드할 수 있습니다.

이 작업 정보

Trident를 제거했다가 다시 설치하는 것은 업그레이드와 같습니다. Trident를 제거해도 Trident 배포에서 사용되는 영구 볼륨 클레임(PVC) 및 영구 볼륨(PV)은 삭제되지 않습니다. 이미 프로비저닝된 PV는 Trident가 오프라인 상태인 동안에도 계속 사용 가능하며, Trident는 온라인 상태로 돌아온 후 그 사이에 생성된 모든 PVC에 대해 볼륨을 프로비저닝합니다.

시작하기 전에

업그레이드하기 전에 "업그레이드 방법을 선택하세요"를 검토한 후 `tridentctl`를 사용하십시오.

단계

1. `tridentctl`에서 제거 명령을 실행하여 CRD 및 관련 개체를 제외한 Trident와 관련된 모든 리소스를 제거합니다.

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

2. Trident를 다시 설치합니다. 다음을 참조하십시오. "tridentctl을 사용하여 Trident를 설치합니다".

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

tridentctl을 사용하여 Trident를 관리합니다

https://github.com/NetApp/trident/releases["Trident 설치 프로그램 번들"^]에는 Trident에 대한 간단한 액세스를 제공하는 `tridentctl` 명령줄 유틸리티가 포함되어 있습니다. 충분한 권한을 가진 Kubernetes 사용자는 이 유틸리티를 사용하여 Trident를 설치하거나 Trident Pod가 포함된 네임스페이스를 관리할 수 있습니다.

명령 및 글로벌 플래그

`tridentctl help`을 실행하여 `tridentctl`에 사용 가능한 명령 목록을 가져오거나 `--help` 플래그를 명령에 추가하여 해당 특정 명령에 대한 옵션 및 플래그 목록을 가져올 수 있습니다.

```
tridentctl [command] [--optional-flag]
```

Trident `tridentctl` 유틸리티는 다음과 같은 명령과 전역 플래그를 지원합니다.

명령

create
Trident에 리소스를 추가합니다.

delete
Trident에서 하나 이상의 리소스를 제거합니다.

get
Trident에서 하나 이상의 리소스를 가져옵니다.

help
모든 명령에 대한 도움말입니다.

images
Trident에 필요한 컨테이너 이미지 테이블을 인쇄합니다.

import
기존 리소스를 Trident로 가져옵니다.

install
Trident를 설치합니다.

logs
Trident에서 로그를 인쇄합니다.

send
Trident에서 리소스를 보냅니다.

uninstall
Trident를 제거합니다.

update
Trident에서 리소스를 수정합니다.

update backend state
백엔드 작업을 일시적으로 일시 중단합니다.

upgrade
Trident에서 리소스를 업그레이드합니다.

version
Trident 버전을 인쇄합니다.

글로벌 플래그

-d, --debug

디버그 출력.

-h, --help

`tridentctl`에 대한 도움말.

-k, --kubeconfig string

`KUBECONFIG` 경로를 지정하여 로컬에서 또는 한 Kubernetes 클러스터에서 다른 클러스터로 명령을 실행합니다.

> 또는 `KUBECONFIG` 변수를 내보내 특정 Kubernetes 클러스터를 가리키도록 한 다음 해당 클러스터에 `tridentctl` 명령을 실행할 수도 있습니다.

-n, --namespace string

Trident 배포의 네임스페이스입니다.

-o, --output string

출력 형식. json|yaml|name|wide|ps 중 하나입니다(기본값).

-s, --server string

Trident REST 인터페이스의 주소/포트.

> Trident REST 인터페이스는 127.0.0.1(IPv4의 경우) 또는 [::1](IPv6의 경우)에서만 수신 대기하고 서비스를 제공하도록 구성할 수 있습니다.

명령 옵션 및 플래그

생성

`create` 명령을 사용하여 Trident에 리소스를 추가합니다.

```
tridentctl create [option]
```

옵션

`backend`: Trident에 백엔드를 추가합니다.

삭제

`delete` 명령을 사용하여 Trident에서 하나 이상의 리소스를 제거합니다.

```
tridentctl delete [option]
```

옵션

backend: Trident에서 하나 이상의 스토리지 백엔드를 삭제합니다.
snapshot: Trident에서 하나 이상의 볼륨 스냅샷을 삭제합니다.
storageclass: Trident에서 하나 이상의 스토리지 클래스를 삭제합니다.
volume: Trident에서 하나 이상의 스토리지 볼륨을 삭제합니다.

가져오기

`get` 명령을 사용하여 Trident에서 하나 이상의 리소스를 가져옵니다.

```
tridentctl get [option]
```

옵션

backend: Trident에서 하나 이상의 스토리지 백엔드를 가져옵니다.
snapshot: Trident에서 하나 이상의 스냅샷을 가져옵니다.
storageclass: Trident에서 하나 이상의 스토리지 클래스를 가져옵니다.
volume: Trident에서 하나 이상의 볼륨을 가져옵니다.

플래그

-h, --help: 볼륨에 대한 도움말.
--parentOfSubordinate string: 하위 소스 볼륨으로 쿼리를 제한합니다.
--subordinateOf string: 볼륨의 하위 볼륨으로 쿼리를 제한합니다.

이미지

`images` 플래그를 사용하여 Trident에 필요한 컨테이너 이미지 테이블을 인쇄합니다.

```
tridentctl images [flags]
```

플래그

-h, --help: 이미지 관련 도움말입니다.
-v, --k8s-version string: Kubernetes 클러스터의 시맨틱 버전입니다.

볼륨 가져오기

`import volume` 명령을 사용하여 기존 볼륨을 Trident로 가져옵니다.

```
tridentctl import volume <backendName> <volumeName> [flags]
```

별칭

volume, v

플래그

-f, --filename string: YAML 또는 JSON PVC 파일 경로.
-h, --help: 볼륨에 대한 도움말.

`--no-manage`: PV/PVC만 생성합니다. 볼륨 수명 주기 관리는 고려하지 않습니다.

설치

`install` 플래그를 사용하여 Trident를 설치합니다.

```
tridentctl install [flags]
```

플래그

`--autosupport-image string`: Autosupport Telemetry용 컨테이너 이미지(기본값 "netapp/trident autosupport:<current-version>").
`--autosupport-proxy string`: Autosupport Telemetry 전송을 위한 프록시 주소/포트.
`--enable-node-prep`: 노드에 필요한 패키지 설치를 시도합니다.
`--generate-custom-yaml`: 아무것도 설치하지 않고 YAML 파일을 생성합니다.
`-h, --help`: 설치에 대한 도움말.
`--http-request-timeout`: Trident 컨트롤러 REST API의 HTTP 요청 시간 초과를 재정의합니다(기본값 1m30s).
`--image-registry string`: 내부 이미지 레지스트리의 주소/포트.
`--k8s-timeout duration`: 모든 Kubernetes 작업의 시간 초과(기본값 3m0s).
`--kubelet-dir string`: kubelet의 내부 상태 호스트 위치(기본값 "/var/lib/kubelet").
`--log-format string`: Trident 로깅 형식(text, json)(기본값 "text").
`--node-prep`: 지정된 데이터 스토리지 프로토콜을 사용하여 볼륨을 관리하도록 Kubernetes 클러스터의 노드를 준비할 수 있도록 Trident를 활성화합니다. 현재 **`iscsi`만 지원됩니다. `OpenShift 4.19`부터 이 기능을 지원하는 최소 `Trident` 버전은 `25.06.1`입니다.**
`--pv string: Trident에서 사용하는 레거시 PV의 이름입니다. 이 이름이 존재하지 않도록 합니다(기본값 "trident").
`--pvc string`: Trident에서 사용하는 레거시 PVC의 이름입니다. 이 이름이 존재하지 않도록 합니다(기본값 "trident").
`--silence-autosupport`: autosupport 번들을 NetApp에 자동으로 보내지 않습니다(기본값 true).
`--silent`: 설치 중 대부분의 출력을 비활성화합니다.
`--trident-image string`: 설치할 Trident 이미지입니다.
`--k8s-api-qps`: Kubernetes API 요청에 대한 초당 쿼리 수(QPS) 제한입니다(기본값 100, 선택 사항).
`--use-custom-yaml`: 설정 디렉터리에 있는 기존 YAML 파일을 사용합니다.
`--use-ipv6`: Trident 통신에 IPv6를 사용합니다.

로그

`logs` 플래그를 사용하여 Trident의 로그를 인쇄합니다.

```
tridentctl logs [flags]
```

플래그

`-a, --archive`: 별도로 지정하지 않는 한 모든 로그를 포함하는 지원 아카이브를 생성합니다.
`-h, --help`: 로그에 대한 도움말입니다.
`-l, --log string`: 표시할 Trident 로그입니다. trident|auto|trident-operator|all 중 하나입니다(기본값 "auto").
`--node string`: 노드 Pod 로그를 수집할 Kubernetes 노드 이름입니다.
`-p, --previous`: 이전 컨테이너 인스턴스가 있는 경우 해당 인스턴스의 로그를 가져옵니다.
`--sidecars`: 사이드카 컨테이너의 로그를 가져옵니다.

전송

`send` 명령을 사용하여 Trident에서 리소스를 전송합니다.

```
tridentctl send [option]
```

옵션

autosupport: NetApp에 Autosupport 아카이브를 보냅니다.

제거

`uninstall` 플래그를 사용하여 Trident를 제거합니다.

```
tridentctl uninstall [flags]
```

플래그

-h, --help: 제거 관련 도움말입니다.
--silent: 제거 과정에서 대부분의 출력을 비활성화합니다.

업데이트

`update` 명령을 사용하여 Trident에서 리소스를 수정합니다.

```
tridentctl update [option]
```

옵션

backend: Trident에서 백엔드를 업데이트합니다.

백엔드 상태 업데이트

`update backend state` 명령을 사용하여 백엔드 작업을 일시 중단하거나 재개합니다.

```
tridentctl update backend state <backend-name> [flag]
```

고려해야 할 사항

- TridentBackendConfig(tbc)를 사용하여 백엔드를 생성한 경우 backend.json 파일을 사용하여 백엔드를 업데이트할 수 없습니다.
- userState`이 tbc에 설정된 경우 `tridentctl update backend state <backend-name> --user-state suspended/normal 명령을 사용하여 수정할 수 없습니다.
- tbc를 통해 설정한 후 tridentctl을 통해 userState`을(를) 다시 설정할 수 있도록 하려면 `userState 필드를 tbc에서 제거해야 합니다. 이는 kubectl edit tbc 명령을 사용하여 수행할 수 있습니다. userState 필드를 제거한 후에는 tridentctl update backend state 명령을 사용하여 백엔드의 `userState`을(를) 변경할 수 있습니다.

- `tridentctl update backend state`를 사용하여 `userState`를 변경합니다. 또한 `userState`를 `TridentBackendConfig` 또는 `backend.json` 파일을 사용하여 업데이트할 수 있습니다. 이 작업은 백엔드의 전체 재초기화를 트리거하며 시간이 오래 걸릴 수 있습니다.

  플래그

  `-h`, `--help`: 백엔드 상태에 대한 도움말입니다.
  `--user-state`: 백엔드 작업을 일시 중지하려면 `suspended`로 설정하십시오. 백엔드 작업을 재개하려면 `normal`로 설정하십시오. `suspended`로 설정하면:

- `AddVolume` 및 `Import Volume`이(가) 일시 중지됩니다.
- `CloneVolume`, `ResizeVolume`, `PublishVolume`, `UnPublishVolume`, `CreateSnapshot`, `GetSnapshot`, `RestoreSnapshot`, `DeleteSnapshot`, `RemoveVolume`, `GetVolumeExternal`, `ReconcileNodeAccess`을(를) 계속 사용할 수 있습니다.

  `userState` 필드를 사용하여 백엔드 구성 파일 `TridentBackendConfig` 또는 `backend.json`에서 백엔드 상태를 업데이트할 수도 있습니다. 자세한 내용은 xref:{relative_path}../trident-use/backend_options.html["백엔드 관리 옵션"] 및 xref:{relative_path}../trident-use/backend_ops_kubectl.html["kubectl을 사용하여 백엔드 관리를 수행합니다"]을 참조하십시오.

예:

JSON

다음 단계를 따라 `userState`를 `backend.json` 파일을 사용하여 업데이트하십시오:

1. `backend.json` 파일을 편집하여 `userState` 필드를 값이 'suspended'로 설정된 상태로 포함하십시오.
2. `tridentctl update backend` 명령어와 업데이트된 `backend.json` 파일의 경로를 사용하여 백엔드를 업데이트하세요.

   예: `tridentctl update backend -f /<path to backend JSON file>/backend.json -n trident`

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "<redacted>",
  "svm": "nas-svm",
  "backendName": "customBackend",
  "username": "<redacted>",
  "password": "<redacted>",
  "userState": "suspended"
}
```

YAML

`kubectl edit <tbc-name> -n <namespace>` 명령을 사용하여 적용된 후 `tbc`를 편집할 수 있습니다. 다음 예에서는 `userState: suspended` 옵션을 사용하여 백엔드 상태를 일시 중단으로 업데이트합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  backendName: customBackend
  storageDriverName: ontap-nas
  managementLIF: <redacted>
  svm: nas-svm
  userState: suspended
  credentials:
    name: backend-tbc-ontap-nas-secret
```

버전

`version` 플래그를 사용하여 `tridentctl` 및 실행 중인 Trident 서비스의 버전을 출력합니다.

```
tridentctl version [flags]
```

플래그

--client: 클라이언트 버전만 해당됩니다(서버 필요 없음).
-h, --help: 버전 관련 도움말.

플러그인 지원

Tridentctl은 kubectl과 유사한 플러그인을 지원합니다. Tridentctl은 플러그인 바이너리 파일 이름이 "tridentctl-<plugin>" 체계를 따르고 바이너리가 PATH 환경 변수에 나열된 폴더에 있는 경우 플러그인을 감지합니다. 감지된 모든 플러그인은 tridentctl help의 플러그인 섹션에 나열됩니다. 선택적으로 환경 변수 TRIDENTCTL_PLUGIN_PATH에서 플러그인 폴더를 지정하여 검색을 제한할 수도 있습니다(예: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). 변수를 사용하는 경우 tridentctl은 지정된 폴더에서만 검색합니다.

Trident 모니터링

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

개요

Trident에서 제공하는 메트릭을 통해 다음과 같은 작업을 수행할 수 있습니다.

- Trident의 상태와 구성을 추적합니다. 작업이 얼마나 성공적인지, 그리고 예상대로 백엔드와 통신할 수 있는지 확인할 수 있습니다.
- 백엔드 사용 정보를 분석하여 백엔드에 프로비저닝된 볼륨 수와 사용된 공간의 양 등을 파악합니다.
- 사용 가능한 백엔드에 프로비저닝된 볼륨 수에 대한 매핑을 유지 관리합니다.
- 성능을 추적하세요. Trident가 백엔드와 통신하고 작업을 수행하는 데 걸리는 시간을 확인할 수 있습니다.

기본적으로 Trident의 메트릭은 대상 포트 `8001`의 `/metrics` 엔드포인트에 노출됩니다. 이러한 메트릭은 Trident 설치 시 *기본적으로 활성화*됩니다. 포트 `8444`에서 HTTPS를 통해 Trident 메트릭을 사용하도록 구성할 수도 있습니다.

필요한 것

- Trident가 설치된 Kubernetes 클러스터.
- Prometheus 인스턴스. 이것은 "컨테이너화된 Prometheus 배포"일 수 있으며, Prometheus를 "네이티브 애플리케이션"로 실행하도록 선택할 수도 있습니다.

1단계: Prometheus 타겟 정의

Trident가 관리하는 백엔드, 생성하는 볼륨 등에 대한 메트릭과 정보를 수집하려면 Prometheus 타겟을 정의해야 합니다. "Prometheus Operator 설명서"을 참조하십시오.

2단계: Prometheus ServiceMonitor 생성

Trident 메트릭을 사용하려면 `trident-csi` 서비스를 감시하고 `metrics` 포트에서 수신 대기하는 Prometheus ServiceMonitor를 생성해야 합니다. 샘플 ServiceMonitor는 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 `trident-csi` 서비스에서 반환되는 메트릭을 검색하고 특히 서비스의 `metrics` 엔드포인트를 찾습니다. 결과적으로 이제 Prometheus는 Trident의 메트릭을 이해하도록 구성되었습니다.

Trident에서 직접 사용할 수 있는 메트릭 외에도 kubelet은 `kubelet_volume_*` 자체 메트릭 엔드포인트를 통해 다양한 메트릭을 제공합니다. Kubelet은 연결된 볼륨, Pod 및 처리하는 기타 내부 작업에 대한 정보를 제공할 수 있습니다. 을 참조하십시오. "여기"

HTTPS를 통해 Trident 메트릭을 사용합니다

HTTPS(포트 8444)를 통해 Trident 메트릭을 사용하려면, ServiceMonitor 정의를 수정하여 TLS 구성을 포함해야 합니다. 또한 `trident-csi` 시크릿을 `trident` 네임스페이스에서 Prometheus가 실행 중인 네임스페이스로 복사해야 합니다. 다음 명령어를 사용하여 이 작업을 수행할 수 있습니다:

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

HTTPS 메트릭을 위한 ServiceMonitor 샘플은 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:
    - interval: 15s
      path: /metrics
      port: https-metrics
      scheme: https
      tlsConfig:
        ca:
          secret:
            key: caCert
            name: trident-csi
        cert:
          secret:
            key: clientCert
            name: trident-csi
        keySecret:
          key: clientKey
          name: trident-csi
        serverName: trident-csi
```

Trident는 tridentctl, Helm 차트 및 Operator를 포함한 모든 설치 방법에서 HTTPS 메트릭을 지원합니다.

- `tridentctl install` 명령을 사용하는 경우 `--https-metrics` 플래그를 전달하여 HTTPS 메트릭을 활성화할 수 있습니다.
- Helm 차트를 사용하는 경우 `httpsMetrics` 매개변수를 설정하여 HTTPS 메트릭을 활성화할 수 있습니다.
- YAML 파일을 사용하는 경우, `--https_metrics` 플래그를 `trident-main` 컨테이너에 `trident-deployment.yaml` 파일에서 추가할 수 있습니다.

3단계: PromQL을 사용하여 Trident 메트릭을 쿼리합니다

PromQL은 시계열 또는 표 형식 데이터를 반환하는 표현식을 생성하는 데 유용합니다.

다음은 사용할 수 있는 PromQL 쿼리입니다.

Trident 상태 정보를 확인하십시오

- **Trident**의 **HTTP 2XX** 응답 비율

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

- 상태 코드별 **Trident**의 **REST** 응답 비율

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

- **Trident**에서 수행한 작업의 평균 기간**(ms)**

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

Trident 사용 정보를 확인하세요

- 평균 볼륨 크기

```
trident_volume_allocated_bytes/trident_volume_count
```

- 각 백엔드에서 프로비저닝한 총 볼륨 공간

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

개별 볼륨 사용량 확인

이 기능은 kubelet 메트릭도 수집되는 경우에만 활성화됩니다.

- 각 볼륨의 사용된 공간 비율

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Trident AutoSupport 원격 측정에 대해 알아보세요

기본적으로 Trident는 Prometheus 메트릭과 기본 백엔드 정보를 매일 NetApp으로 전송합니다.

- Trident가 Prometheus 메트릭 및 기본 백엔드 정보를 NetApp으로 전송하지 못하도록 하려면 Trident 설치 중에 `--silence-autosupport` 플래그를 전달하십시오.
- Trident는 `tridentctl send autosupport`를 통해 필요에 따라 컨테이너 로그를 NetApp 지원팀에 전송할 수도 있습니다. Trident가 로그를 업로드하도록 트리거해야 합니다. 로그를 제출하기 전에 NetApp 의https://www.netapp.com/company/legal/privacy-policy/["개인정보 보호정책"^]에 동의해야 합니다.
- 별도로 지정하지 않으면 Trident는 지난 24시간 동안의 로그를 가져옵니다.
- `--since` 플래그를 사용하여 로그 보존 기간을 지정할 수 있습니다. 예를 들면 다음과 같습니다: `tridentctl send autosupport --since=1h`. 이 정보는 Trident와 함께 설치되는 `trident-autosupport` 컨테이너를 통해 수집되고 전송됩니다. 컨테이너 이미지는 "Trident AutoSupport"에서 얻을 수 있습니다.
- Trident AutoSupport는 개인 식별 정보(PII) 또는 개인 정보를 수집하거나 전송하지 않습니다. Trident 컨테이너 이미지 자체에는 적용되지 않는 "EULA"와 함께 제공됩니다. NetApp의 데이터 보안 및 신뢰에 대한 노력에 대해 자세히 알아볼 수 있습니다 "여기".

Trident에서 보낸 페이로드의 예는 다음과 같습니다.

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

- AutoSupport 메시지는 NetApp의 AutoSupport 엔드포인트로 전송됩니다. 컨테이너 이미지를 저장하는 데 개인 레지스트리를 사용하는 경우 `--image-registry` 플래그를 사용할 수 있습니다.
- 설치 YAML 파일을 생성하여 프록시 URL을 구성할 수도 있습니다. 이 작업은 YAML 파일을 생성하기 위해 `tridentctl install --generate-custom-yaml`를 사용하고, `trident-autosupport` 컨테이너에 `--proxy-url` 인수를 `trident-deployment.yaml`에 추가하여 수행할 수 있습니다.

Trident 메트릭을 비활성화합니다

메트릭 보고를 비활성화하려면, 사용자 지정 YAML을 생성해야 합니다(`--generate-custom-yaml` 플래그 사용). 그런 다음 해당 파일을 편집하여 `--metrics` 플래그가 `trident-main` 컨테이너에서 호출되지 않도록 제거해야 합니다.

Trident 제거

Trident를 설치하는 데 사용한 것과 동일한 방법을 사용하여 Trident를 제거해야 합니다.

이 작업 정보

- 업그레이드 후 발견된 버그, 종속성 문제 또는 업그레이드 실패나 불완전한 업그레이드에 대한 수정이 필요한 경우, Trident를 제거하고 해당 "버전"에 대한 특정 지침을 사용하여 이전 버전을 다시 설치해야 합니다. 이것이 이전 버전으로 _다운그레이드_하는 유일하게 권장되는 방법입니다.
- 간편한 업그레이드 및 재설치를 위해 Trident를 제거해도 Trident에서 생성한 CRD 또는 관련 객체는 제거되지 않습니다. Trident와 모든 데이터를 완전히 제거해야 하는 경우 "Trident 및 CRD를 완전히 제거합니다"를 참조하십시오.

시작하기 전에

Kubernetes 클러스터를 사용 중지하는 경우 제거하기 전에 Trident에서 생성한 볼륨을 사용하는 모든 애플리케이션을 삭제해야 합니다. 이렇게 하면 PVC가 삭제되기 전에 Kubernetes 노드에서 게시 취소됩니다.

원래 설치 방법을 확인하십시오

Trident를 설치할 때 사용했던 방법과 동일한 방법으로 Trident를 제거해야 합니다. 제거하기 전에 원래 Trident를 설치하는 데 사용한 버전을 확인하십시오.

1. `kubectl get pods -n trident`를 사용하여 Pod를 검사합니다.
   - 오퍼레이터 포드가 없는 경우 Trident가 `tridentctl`을 사용하여 설치되었습니다.
   - 오퍼레이터 포드가 있는 경우, Trident는 수동으로 또는 Helm을 사용하여 Trident 오퍼레이터를 통해 설치되었습니다.
2. 운영자 포드가 있는 경우 `kubectl describe tproc trident`를 사용하여 Helm을 통해 Trident가 설치되었는지 확인합니다.
   - Helm 레이블이 있는 경우 Trident는 Helm을 사용하여 설치되었습니다.
   - Helm 레이블이 없으면 Trident 운영자를 사용하여 Trident를 수동으로 설치한 것입니다.

Trident 운영자 설치를 제거합니다

Trident Operator 설치는 수동으로 또는 Helm을 사용하여 제거할 수 있습니다.

수동 설치 제거

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

1. **CR `TridentOrchestrator`**을 편집하고 제거 플래그를 설정하세요:

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

`uninstall` 플래그가 `true`로 설정되면 Trident 운영자는 Trident를 제거하지만 TridentOrchestrator 자체는 제거하지 않습니다. Trident를 다시 설치하려면 TridentOrchestrator를 정리하고 새로 생성해야 합니다.

2. 삭제 **TridentOrchestrator**: Trident를 배포하는 데 사용된 TridentOrchestrator CR을 제거하면 운영자에게 Trident를 제거하도록 지시합니다. 운영자는 TridentOrchestrator 제거를 처리하고 Trident 배포 및 데몬셋을 제거하며, 설치 과정에서 생성한 Trident Pod를 삭제합니다.

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

Helm 설치 제거

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

```
#List the Helm release corresponding to the 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
```

tridentctl 설치 제거

Trident와 관련된 CRD 및 관련 오브젝트를 제외한 모든 리소스를 삭제하려면 uninstall 명령을 `tridentctl`에서 사용하십시오.

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