Trident 관리 및 모니터링

Trident

NetApp
February 02, 2026

목차

Trident 관리 및 모니터링

Trident를 업그레이드합니다

Trident를 업그레이드합니다

24.02 릴리즈부터 Trident는 4개월 릴리즈 주기를 따르며 매년 3개의 주요 릴리즈를 제공합니다. 각각의 새 릴리스는 이전 릴리스에서 빌드되며 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 릴리즈로 업그레이드할 때 다음 사항을 고려하십시오.

- 특정 Kubernetes 클러스터의 모든 네임스페이스에 Trident 인스턴스는 하나만 설치해야 합니다.
- Trident 23.07 이상에서는 v1 볼륨 스냅샷이 필요하며 알파 또는 베타 스냅샷을 더 이상 지원하지 않습니다.
- 업그레이드할 때 Trident에서 사용되는 를 `StorageClasses` 제공하는 것이 중요합니다 `parameter.fsType`. 기존 볼륨을 중단하지 않고 삭제 및 재생성할 수 `StorageClasses` 있습니다.
  - 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

Trident 버전은 날짜 기반 `YY.MM` 명명 규칙을 따릅니다. 여기서 "YY"는 연도의 마지막 두 자리이고 "MM"은 월입니다. DOT 릴리스는 `YY.MM.X` 규칙을 따릅니다. 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 24.06(또는 24.06 도트 릴리스)에서 25.06로 직접 업그레이드할 수 있습니다.
- 릴리스 4개가 아닌 다른 릴리스에서 업그레이드하는 경우 여러 단계로 업그레이드를 수행합니다. 에서 4개의 릴리스 윈도우에 맞는 가장 최신 릴리스로 업그레이드하려는 의 업그레이드 지침을 "이전 버전" 사용합니다. 예를 들어, 23.07을 실행하고 있고 25.06로 업그레이드하려는 경우:
  a. 23.07에서 24.06로 첫 번째 업그레이드.
  b. 그런 다음 24.06에서 25.06로 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

Trident를 처음 설치할 때 사용한 버전을 확인하려면 다음을 수행합니다.

1. 사용 `kubectl get pods -n trident` 를 눌러 포드를 검사합니다.
   - 운영자 포드가 없으면 를 사용하여 Trident를 설치한 `tridentctl` 것입니다.
   - 운영자 포드가 있는 경우, Trident는 Trident 운영자를 사용하여 수동으로 또는 Helm을 사용하여 설치되었습니다.
2. 운영자 포드가 있는 경우 를 `kubectl describe torc` 사용하여 Trident가 Helm을 사용하여 설치되었는지 확인합니다.
   - Helm 레이블이 있는 경우 Helm을 사용하여 Trident를 설치한 것입니다.
   - Helm 레이블이 없는 경우 Trident 연산자를 사용하여 Trident를 수동으로 설치했습니다.

3단계**:** 업그레이드 방법을 선택합니다

일반적으로 초기 설치에 사용한 것과 동일한 방법으로 업그레이드해야 하지만, 가능한 경우"설치 방법 간에 이동합니다" 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 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

Trident를 설치하고 업그레이드할 수 있는 툴 중 "Trident 연산자를 사용할 때의 이점"하나는 기존의 마운트된 볼륨을 중단하지 않고 Trident 및 Kubernetes 개체를 자동으로 처리하는 것입니다. 이와 같이 Trident는 다운타임 없이 업그레이드를 지원할 수 있습니다. 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 Trident 아키텍처에 대한 자세한 내용은 을 참조하십시오"Trident 아키텍처".

운영자 업그레이드 워크플로우

Trident 연산자를 사용하여 업그레이드를 시작하는 경우:

1. Trident 운영자 *:
   a. 현재 설치된 Trident 버전(version_n_)을 검색합니다.
   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 operator 또는 **Helm**을 사용하여 **Trident** 설치를 업그레이드합니다

Trident 연산자를 사용하여 수동으로 또는 Helm을 사용하여 Trident를 업그레이드할 수 있습니다. Trident 운영자 설치에서 다른 Trident 운영자 설치로 업그레이드하거나, 설치에서 Trident 운영자 버전으로 업그레이드할 수 `tridentctl` 있습니다. "업그레이드 방법을 선택합니다"Trident 운영자 설치를 업그레이드하기 전에 검토하십시오.

수동 설치를 업그레이드합니다

클러스터 범위 Trident Operator 설치에서 다른 클러스터 범위 Trident Operator 설치로 업그레이드할 수 있습니다. 모든 Trident 버전은 클러스터 범위 연산자를 사용합니다.

> ⓘ 네임스페이스 범위 연산자(버전 20.07 ~ 20.10)를 사용하여 설치된 Trident에서 업그레이드하려면 Trident의 업그레이드 지침을 "설치된 버전"사용하십시오.

이 작업에 대해

Trident는 운영자를 설치하고 Kubernetes 버전에 대한 관련 오브젝트를 생성하는 데 사용할 수 있는 번들 파일을 제공합니다.

- Kubernetes 1.24를 실행하는 클러스터에는 "Bundle_PRE_1_25.YAML"를 사용합니다.
- Kubernetes 1.25 이상을 실행하는 클러스터의 경우 를 "Bundle_post_1_25.YAML"사용합니다.

시작하기 전에

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

단계

1. Trident 버전 확인:

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

2. 업데이트 `operator.yaml` , `tridentorchestrator_cr.yaml` , 그리고 `post_1_25_bundle.yaml` 업그레이드하려는 버전(예: 25.06)에 대한 레지스트리와 이미지 경로, 그리고 올바른 비밀번호를 사용합니다.
3. 현재 Trident 인스턴스를 설치하는 데 사용된 Trident 연산자를 삭제합니다. 예를 들어, 25.02에서 업그레이드하는 경우 다음 명령을 실행합니다.

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

4. 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.
5. _<bundle.yaml>_이 있는 환경에 맞는 올바른 번들 YAML 파일을 사용하여 Trident 설치하세요. `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml` Kubernetes 버전에 따라 다릅니다. 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. 트라이던트 토크를 편집하여 이미지 25.06.0을 포함시킵니다.

Helm 설치를 업그레이드합니다

Trident Helm 설치를 업그레이드할 수 있습니다.

> Trident가 설치된 Kubernetes 클러스터를 1.24에서 1.25 이상으로 업그레이드할 `true` `helm upgrade` 경우 클러스터를 업그레이드하기 전에 value.yaml을 명령으로 설정하거나 `--set excludePodSecurityPolicy=true` 명령에 추가해야 `excludePodSecurityPolicy` 합니다.

Trident Helm을 업그레이드하지 않고 이미 Kubernetes 클러스터를 1.24에서 1.25로 업그레이드한 경우에는 Helm 업그레이드가 실패합니다. Helm 업그레이드를 진행하려면 다음 단계를 전제 조건으로 수행하십시오.

1. 에서 helm-mapkubeapis 플러그인을 https://github.com/helm/helm-mapkubeapis 설치합니다.
2. Trident가 설치된 네임스페이스에서 Trident 릴리스에 대해 건식 실행을 수행합니다. 정리할 리소스가 나열됩니다.

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

3. 정리 작업을 수행하려면 Helm을 사용하여 전체 실행을 수행합니다.

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

단계

1. "Helm을 사용하여 Trident를 설치했습니다"를 사용하여 한 단계로 업그레이드할 수 `helm upgrade trident netapp-trident/trident-operator --version 100.2506.0` 있습니다. Helm repo를 추가하지 않았거나 업그레이드에 사용할 수 없는 경우:

   a. 에서 최신 Trident 릴리스를 "GitHub의 _Assets_섹션"다운로드하십시오.

   b. 사용하다 `helm upgrade` 명령은 어디에 `trident-operator-25.10.0.tgz` 업그레이드하려는 버전을 나타냅니다.

   ```
   helm upgrade <name> trident-operator-25.10.0.tgz
   ```

   초기 설치 중에 사용자 정의 옵션을 설정하는 경우(예: Trident 및 CSI 이미지에 대한 전용 미러링된 레지스트리 지정) 를 추가합니다 `helm upgrade` 명령을 사용합니다 `--set` 이러한 옵션이 업그레이드 명령에 포함되도록 하려면 값이 기본값으로 재설정됩니다.

2. 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 Trident 릴리스를 다운로드합니다.

   ```
   # Download the release required [25.10.0]
   mkdir 25.10.0
   cd 25.10.0
   wget
   https://github.com/NetApp/trident/releases/download/v25.10.0/trident-
   installer-25.10.0.tar.gz
   tar -xf trident-installer-25.10.0.tar.gz
   cd trident-installer
   ```

2. 매니페스트에서 트라이디오케스트레이터 CRD를 만듭니다.

   ```
   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. `TridentOrchestrator`Trident 설치를 위한 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 확인합니다.

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

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v25.10.0
```

tridentctl로 업그레이드하십시오

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

이 작업에 대해

Trident 제거 및 재설치는 업그레이드로 작동합니다. Trident를 제거하면 Trident 배포에 사용되는 PVC(영구 볼륨 클레임) 및 PV(영구 볼륨)가 삭제되지 않습니다. 이미 프로비저닝된 PVS는 Trident가 오프라인 상태인 동안 사용 가능한 상태로 유지되며, Trident는 다시 온라인 상태가 된 후 그 사이에 생성된 모든 PVC에 대한 볼륨을 프로비저닝합니다.

시작하기 전에

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

단계

1. 에서 uninstall 명령을 `tridentctl` 실행하여 CRD 및 관련 객체를 제외한 Trident와 연결된 모든 리소스를 제거합니다.

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

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

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

tridentctl을 사용하여 Trident를 관리합니다

에는 "Trident 설치 프로그램 번들" Trident에 간단하게 액세스할 수 있는 명령줄 유틸리티가 포함되어 `tridentctl` 있습니다. 충분한 Privileges를 가진 Kubernetes 사용자는 이 툴을 사용하여 Trident를 설치하거나 Trident Pod가 포함된 네임스페이스를 관리할 수 있습니다.

명령 및 글로벌 플래그

실행할 수 있습니다 `tridentctl help` 에 사용할 수 있는 명령 목록을 가져옵니다 `tridentctl` 또는 를 추가합니다 `--help` 특정 명령에 대한 옵션 및 플래그 목록을 가져오려면 임의의 명령에 플래그를 지정합니다.

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

Trident `tridentctl` 유틸리티는 다음 명령 및 글로벌 플래그를 지원합니다.

명령

create
Trident에 리소스를 추가합니다.

delete
Trident에서 하나 이상의 리소스를 제거합니다.

get
Trident에서 하나 이상의 리소스를 얻습니다.

help
모든 명령에 대한 도움말.

images
Trident에 필요한 컨테이너 이미지 표를 인쇄합니다.

import
기존 리소스를 Trident로 가져옵니다.

install
Trident를 설치합니다.

logs
Trident에서 로그를 인쇄합니다.

send
Trident에서 리소스를 보냅니다.

"제거"를 선택합니다
Trident를 제거합니다.

update
Trident에서 리소스를 수정합니다.

update backend state
백엔드 작업을 일시적으로 중단합니다.

upgrade
Trident에서 리소스를 업그레이드합니다.

'내전'
Trident 버전을 인쇄합니다.

글로벌 플래그

-d, --debug

디버그 출력.

-h, --help

도움말 `tridentctl`.

-k, --kubeconfig string

를 지정합니다 `KUBECONFIG` 로컬로 또는 Kubernetes 클러스터 간에 명령을 실행할 수 있는 경로입니다.

또는 를 내보낼 수 있습니다 `KUBECONFIG` 특정 Kubernetes 클러스터 및 문제를 가리키는 변수 `tridentctl` 명령을 제공할 수 있습니다.

-n, --namespace string

Trident 배포의 네임스페이스입니다.

-o, --output string

출력 형식. json|YAML|name|wide|ps(기본값) 중 하나.

-s, --server string

Trident REST 인터페이스의 주소/포트입니다.

Trident REST 인터페이스는 127.0.0.1(IPv4의 경우) 또는 [::1](IPv6의 경우)에서만 수신 및 서비스하도록 구성할 수 있습니다.

명령 옵션 및 플래그

생성

명령을 사용하여 `create` Trident에 리소스를 추가할 수 있습니다.

```
tridentctl create [option]
```

옵션

`backend`: Trident에 백엔드를 추가합니다.

삭제

명령을 사용하여 `delete` Trident에서 하나 이상의 리소스를 제거할 수 있습니다.

```
tridentctl delete [option]
```

옵션

`backend`: Trident에서 하나 이상의 저장소 백엔드를 삭제합니다.
`snapshot`: Trident에서 하나 이상의 볼륨 스냅샷을 삭제합니다.
`storageclass`: Trident에서 하나 이상의 저장소 클래스를 삭제합니다.

`volume`: Trident에서 하나 이상의 저장소 볼륨을 삭제합니다.

가져오기

명령을 사용하여 `get` 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.

깃발

`-h, --help`: 볼륨에 대한 도움말입니다.
`--parentOfSubordinate string`: 하위 원본 볼륨으로 쿼리를 제한합니다.
`--subordinateOf string`: 볼륨 부하로 쿼리 제한.

이미지

`images`플래그를 사용하여 `Trident`에 필요한 컨테이너 이미지 테이블을 인쇄합니다.

```
tridentctl images [flags]
```

깃발

`-h, --help`이미지 도움말.
`-v, --k8s-version string`: Kubernetes 클러스터의 시맨틱 버전입니다.

볼륨 가져오기

명령을 사용하여 `import volume` 기존 볼륨을 Trident로 가져옵니다.

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

별칭

`volume, v`

깃발

`-f, --filename string`YAML 또는 `JSON PVC` 파일로 이동합니다.
`-h, --help`: 볼륨에 대한 도움말입니다.
`--no-manage`:PV/PVC만 생성 볼륨 라이프사이클 관리를 가정하지 마십시오.

설치합니다

`install`플래그를 사용하여 `Trident`를 설치합니다.

```
tridentctl install [flags]
```

깃발

- `--autosupport-image string`: Autosupport Telemetry의 컨테이너 이미지(기본값 "netapp/trident autosupport:<현재 버전>").
- `--autosupport-proxy string` : Autosupport Telemetry를 보내기 위한 프록시의 주소/포트입니다.
- `--enable-node-prep` : 노드에 필요한 패키지를 설치해 보세요.
- `--generate-custom-yaml` : 아무것도 설치하지 않고 YAML 파일을 생성합니다.
- `-h` , `--help` : 설치에 대한 도움말.
- `--http-request-timeout` : Trident 컨트롤러의 REST API에 대한 HTTP 요청 시간 초과를 재정의합니다 (기본값 1분 30초).
- `--image-registry string` : 내부 이미지 레지스트리의 주소/포트.
- `--k8s-timeout duration` : 모든 Kubernetes 작업에 대한 시간 초과(기본값 3분 0초).
- `--kubelet-dir string` : kubelet의 내부 상태의 호스트 위치(기본값 "/var/lib/kubelet").
- `--log-format string` : Trident 로깅 형식(텍스트, JSON)(기본값은 "텍스트")입니다.
- `--node-prep` : Trident 지정된 데이터 저장 프로토콜을 사용하여 볼륨을 관리하도록 Kubernetes 클러스터의 노드를 준비할 수 있도록 합니다. 현재, **`iscsi`** 지원되는 유일한 값입니다**. OpenShift 4.19**부터 이 기능을 지원하는 최소 **Trident** 버전은 **25.06.1**입니다**.**
- `--pv string` : Trident 에서 사용되는 레거시 PV의 이름으로, 이것이 존재하지 않도록 합니다(기본값은 "trident").
- `--pvc string` : Trident 에서 사용하는 레거시 PVC의 이름으로, 이것이 존재하지 않도록 합니다(기본값은 "trident").
- `--silence-autosupport` : NetApp 에 자동 지원 번들을 자동으로 보내지 않습니다(기본값 true).
- `--silent` : 설치 중에 대부분의 출력을 비활성화합니다.
- `--trident-image string` : 설치할 Trident 이미지입니다.
- `--k8s-api-qps` : Kubernetes API 요청에 대한 초당 쿼리 수(QPS) 제한(기본값 100, 선택 사항).
- `--use-custom-yaml` : 설치 디렉토리에 있는 기존 YAML 파일을 사용합니다.
- `--use-ipv6` : Trident 통신에 IPv6를 사용합니다.

로그

`logs`플래그를 사용하여 `Trident`에서 로그를 인쇄합니다.

```
tridentctl logs [flags]
```

깃발

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

전송

명령을 사용하여 `send` Trident에서 리소스를 보냅니다.

```
tridentctl send [option]
```

옵션

`autosupport`AutoSupport 아카이브를 NetApp으로 전송합니다.

설치 제거

`uninstall`플래그를 사용하여 `Trident`를 제거합니다.

```
tridentctl uninstall [flags]
```

깃발

`-h, --help`: 제거 도움말입니다.
`--silent`: 제거 중 대부분의 출력을 비활성화합니다.

업데이트

명령을 사용하여 `update` Trident에서 리소스를 수정합니다.

```
tridentctl update [option]
```

옵션

`backend`: Trident에서 백엔드를 업데이트합니다.

백엔드 상태를 업데이트합니다

를 사용합니다 `update backend state` 백엔드 작업을 일시 중지하거나 재개하는 명령입니다.

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

고려해야 할 사항

- TridentBackendConfig(tbc)를 사용하여 백엔드를 생성한 경우 파일을 사용하여 백엔드를 업데이트할 수 `backend.json` 없습니다.
- 가 tbc에 설정된 경우 `userState` 명령을 사용하여 수정할 수 없습니다 `tridentctl update backend state <backend-name> --user-state suspended/normal` .
- tbc를 통해 설정한 후 via tridentctl을 `userState` 다시 설정하려면 `userState` tbc에서 필드를 제거해야 합니다. 이 작업은 명령을 사용하여 수행할 수 `kubectl edit tbc` 있습니다. 필드가 제거된 후 `userState` 명령을 사용하여 백엔드의 을 변경할 `userState` 수 있습니다 `tridentctl update backend state`.
- 를 사용하여 `tridentctl update backend state` 를 `userState`변경합니다. 또는 파일을 사용하여 업데이트할 수도 `userState` `TridentBackendConfig` `backend.json` 있습니다. 이렇게 하면 백엔드의 완전한 재초기화가 트리거되고 시간이 오래 걸릴 수 있습니다.

  깃발

  `-h, --help`: 백엔드 상태에 대한 도움말입니다.
  `--user-state`: 로 설정합니다 `suspended` 백엔드 작업을 일시 중지합니다. 를 로 설정합니다 `normal` 백엔드 작업을 재개합니다. 를 로 설정한 경우 `suspended`:

- `AddVolume` 그리고 `Import Volume` 일시 중지되었습니다.

- `CloneVolume` `ResizeVolume`, , `PublishVolume`, , , , `UnPublishVolume` `CreateSnapshot` `GetSnapshot` `RestoreSnapshot`, , `DeleteSnapshot`, , , , `RemoveVolume` `GetVolumeExternal` `ReconcileNodeAccess` 사용 가능 상태를 유지합니다.

백엔드 구성 파일 또는 의 필드를 사용하여 백엔드 상태를 업데이트할 수도 `userState` `TridentBackendConfig` `backend.json`있습니다. 자세한 내용은 및 을 "백엔드 관리 옵션" "kubeck을 사용하여 백엔드 관리 수행"참조하십시오.

- 예: *

JSON을 참조하십시오

파일을 사용하여 를 업데이트하려면 다음 단계를 `userState` `backend.json` 수행하십시오.

1. `backend.json`값이 `'uspended'`로 설정된 필드를 포함하도록 파일을 `userState` 편집합니다.
2. 백엔드를 업데이트하려면 다음을 사용하세요. `tridentctl update backend` 명령 및 업데이트 경로 `backend.json` 파일.

   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

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

```
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

사용 `version` 플래그를 사용하여 의 버전을 인쇄합니다 `tridentctl` 및 실행 중인 Trident 서비스를 제공합니다.

```
tridentctl version [flags]
```

깃발

`--client`: 클라이언트 버전만(서버가 필요하지 않음).
`-h, --help`: 버전에 대한 도움말입니다.

플러그인 지원

Tridentctl은 kubectl과 유사한 플러그인을 지원합니다. Tridentctl은 플러그인 바이너리 파일 이름이 "tridentctl-<plugin>" 체계를 따르고 바이너리가 경로 환경 변수를 나열한 폴더에 있는 경우 플러그인을 감지합니다. 검색된 모든 플러그인은 tridentctl 도움말의 플러그인 섹션에 나열됩니다. 필요한 경우 환경 변수 TRIDENTCTL_PLUGIN_PATH에 플러그인 폴더를 지정하여 검색을 제한할 수도 있습니다(예: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). 변수가 사용되는 경우, tridenctl 은 지정된 폴더에서만 검색합니다.

Trident 모니터링

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

개요

Trident에서 제공하는 메트릭을 사용하면 다음을 수행할 수 있습니다.

- Trident의 상태 및 구성을 지속적으로 확인합니다. 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. Trident가 백엔드와 통신하고 작업을 수행하는 데 걸리는 시간을 살펴볼 수 있습니다.

> 기본적으로 Trident의 메트릭은 대상 포트에 노출됩니다. `8001` 에서 `/metrics` 종점. 이러한 측정항목은 Trident 가 설치되면 기본적으로 활성화됩니다. HTTPS를 통해 포트에서 Trident 메트릭을 사용하도록 구성할 수 있습니다. `8444` 또한.

필요한 것

- 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

Prometheus 대상을 정의하여 Trident 관리하는 백엔드, Trident가 생성하는 볼륨 등에 대한 메트릭과 정보를 수집해야 합니다. 보다"Prometheus Operator 문서" .

2단계: Prometheus ServiceMonitor를 만듭니다

Trident 메트릭을 소비하려면 트리덴트 CSI 서비스를 감시하고 메티우스 포트에서 수신 대기하는 프로메테우스 서비스

모니터를 만들어야 합니다. 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 다음에 의해 반환된 메트릭을 검색합니다. `trident-csi` 서비스를 특별히 찾습니다 `metrics` 서비스의 종료점. 결과적으로 Prometheus는 이제 Trident의 측정 항목을 이해하도록 구성되었습니다.

kubelet은 Trident에서 직접 사용할 수 있는 메트릭과 더불어 자체 메트릭 엔드포인트를 통해 많은 메트릭을 노출합니다. `kubelet_volume_*` Kubelet는 연결된 볼륨, Pod 및 처리하는 기타 내부 작업에 대한 정보를 제공할 수 있습니다. 을 "여기"참조하십시오.

HTTPS를 통해 **Trident** 메트릭 사용

HTTPS(포트 8444)를 통해 Trident 메트릭을 사용하려면 TLS 구성을 포함하도록 ServiceMonitor 정의를 수정해야 합니다. 또한 다음을 복사해야 합니다. `trident-csi` 의 비밀 `trident` Prometheus가 실행되는 네임스페이스에 대한 네임스페이스입니다. 다음 명령을 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 메트릭에 대한 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
```

Trident tridentctl, Helm 차트, Operator 등 모든 설치 방법에서 HTTPS 메트릭을 지원합니다.

- 당신이 사용하는 경우 `tridentctl install` 명령을 전달하면 `--https-metrics` HTTPS 메트릭을 활성화하는 플래그입니다.
- Helm 차트를 사용하는 경우 다음을 설정할 수 있습니다. `httpsMetrics` HTTPS 메트릭을 활성화하는 매개변수입니다.
- YAML 파일을 사용하는 경우 다음을 추가할 수 있습니다. `--https_metrics` 에 깃발을 `trident-main` 컨테이너에 `trident-deployment.yaml` 파일.

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

- **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 코드를 통한 **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 지속 시간**(ms)**

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

Trident 사용 정보를 가져옵니다

- 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
```

Trident AutoSupport 원격 측정 기능에 대해 알아봅니다

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

- Trident에서 Prometheus 메트릭 및 기본 백엔드 정보를 NetApp로 보내지 않도록 하려면 `--silence -autosupport` Trident 설치 중에 플래그를 전달합니다.
- 또한 Trident는 를 통해 컨테이너 로그를 NetApp 지원 팀에 온디맨드로 전송할 수 `tridentctl send autosupport` 있습니다. 로그를 업로드하려면 Trident를 트리거해야 합니다. 로그를 제출하기 전에 NetApp를 수락해야 "개인 정보 보호 정책"합니다.
- 별도로 지정하지 않는 한 Trident는 지난 24시간의 로그를 가져옵니다.
- 플래그를 사용하여 로그 보존 기간을 지정할 수 `--since` 있습니다. 예를 들면 다음과 `tridentctl send autosupport --since=1h`같습니다. 이 정보는 `Trident`와 함께 설치된 컨테이너를 통해 수집되고 전송됩니다 `trident-autosupport`. 컨테이너 이미지는 에서 얻을 수 "Trident AutoSupport를 누릅니다" 있습니다.
- Trident AutoSupport는 개인 식별 정보(PII) 또는 개인 정보를 수집하거나 전송하지 않습니다. 에는 Trident 컨테이너 이미지 자체에 적용할 수 없는 가 "EULA" 포함되어 있습니다. 데이터 보안 및 신뢰에 대한 NetApp의 노력에 대해 더 자세히 알아볼 수 "여기"있습니다.

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. 개인 레지스트리를 사용하여 컨테이너 이미지를 저장하는 경우 '--image-registry' 플래그를 사용할 수 있습니다.
- 또한 설치 YAML 파일을 생성하여 프록시 URL을 구성할 수도 있습니다. 이는 트라이덴트ctl install --generate -custom-YAML을 이용해 YAML 파일을 생성하고 트리덴트 배포(trident-deployment)의 트리덴트 자동 지원 컨테이너에 대한 '--proxy-url' 주장을 추가하는 방식으로 가능하다.

Trident 메트릭을 비활성화합니다

- 메트릭을 보고하지 않으려면 ('--generate-custom-YAML' 플래그를 사용하여) 사용자 지정 YAML을 생성하고 이를 편집하여 삼중류-main' 컨테이너에 대해 호출되는 '--metrics' 플래그를 제거해야 합니다.

Trident를 제거합니다

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

이 작업에 대해

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

시작하기 전에

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

원래 설치 방법을 확인합니다

Trident를 설치할 때 사용한 것과 동일한 방법을 사용해야 합니다. 제거하기 전에 Trident를 처음 설치할 때 사용한 버전을 확인하십시오.

1. 사용 `kubectl get pods -n trident` 를 눌러 포드를 검사합니다.
   - 운영자 포드가 없으면 를 사용하여 Trident를 설치한 `tridentctl` 것입니다.
   - 운영자 포드가 있는 경우, Trident는 Trident 운영자를 사용하여 수동으로 또는 Helm을 사용하여 설치되었습니다.
2. 운영자 포드가 있는 경우 를 `kubectl describe tproc trident` 사용하여 Trident가 Helm을 사용하여 설치되었는지 확인합니다.
   - Helm 레이블이 있는 경우 Helm을 사용하여 Trident를 설치한 것입니다.
   - Helm 레이블이 없는 경우 Trident 연산자를 사용하여 Trident를 수동으로 설치했습니다.

Trident 운영자 설치를 제거합니다

수동 또는 Helm을 사용하여 트라이덴트 작업자 설치를 제거할 수 있습니다.

수동 설치를 제거합니다

운영자를 사용하여 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`**: Trident를 배포하는 데 사용된 CR을 제거하면 `TridentOrchestrator` 운영자에게 Trident를 제거하도록 지시합니다. 운영자는 의 제거를 `TridentOrchestrator` 처리하고 Trident 배포 및 데몬 세트를 제거하며 설치의 일부로 생성한 Trident 포드를 삭제합니다.

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

Helm 설치를 제거합니다

Helm을 사용하여 Trident를 설치한 경우 을 사용하여 제거할 수 `helm uninstall` 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
```

를 제거합니다 `tridentctl` 설치

`uninstall`CRD 및 관련 객체를 제외하고 `Trident`와 연결된 모든 리소스를 제거하려면 의 명령을 `tridentctl` 사용합니다.

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