



Trident 관리 및 모니터링

Trident

NetApp
September 26, 2025

목차

Trident 관리 및 모니터링	1
Trident를 업그레이드합니다	1
Trident를 업그레이드합니다	1
운영자와 함께 업그레이드하십시오	2
tridentctl로 업그레이드하십시오	7
tridentctl을 사용하여 Trident를 관리합니다	7
명령 및 글로벌 플래그	7
명령 옵션 및 플래그	9
플러그인 지원	14
Trident 모니터링	14
개요	14
1단계: Prometheus 목표를 정의합니다	14
2단계: Prometheus ServiceMonitor를 만듭니다	15
3단계: PromQL을 사용하여 Trident 메트릭 쿼리	15
Trident AutoSupport 원격 측정 기능에 대해 알아봅니다	16
Trident 메트릭을 비활성화합니다	17
Trident를 제거합니다	17
원래 설치 방법을 확인합니다	18
Trident 운영자 설치를 제거합니다	18
설치를 제거합니다 tridentctl	19

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.
- 에서 Google Cloud용 Cloud Volumes Service를 생성한 경우 Trident 23.01에서 업그레이드할 때 또는 zoneredundantstandardsw 서비스 수준을 "CVS 서비스 유형입니다" 사용하도록 백엔드 구성을 업데이트해야 standardsw 합니다. 백엔드에서 을 업데이트하지 serviceLevel 못하면 볼륨이 실패할 수 있습니다. 자세한 내용은 을 "CVS 서비스 유형 샘플" 참조하십시오.
- 업그레이드할 때 Trident에서 사용되는 를 StorageClasses 제공하는 것이 중요합니다 parameter.fsType. 기존 볼륨을 중단하지 않고 삭제 및 재생성할 수 StorageClasses 있습니다.
 - 이는 SAN 볼륨에 적용할 때 ** 요구 "보안 컨텍스트" 사항입니다.
 - [https://github.com/NetApp/TridentTrident/tree/master/TridentTrident-installer/sample-input\[sample input\]](https://github.com/NetApp/TridentTrident/tree/master/TridentTrident-installer/sample-input[sample input]) 디렉터리에는 [https://github.com/NetApp/Trident/blob/master/Trident-installer/sample/storage-class-\[storage-class-bronze-default.yaml\]basic.yaml\[storage-class-basic.yaml.template\]](https://github.com/NetApp/Trident/blob/master/Trident-installer/sample/storage-class-[storage-class-bronze-default.yaml]basic.yaml[storage-class-basic.yaml.template])
 - 자세한 내용은 을 "알려진 문제"참조하십시오.

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 23.04(또는 23.04 도트 릴리스)에서 24.06로 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, 22.01을 실행하고 있고 24.06로 업그레이드하려는 경우:
 - a. 22.07에서 23.04로 첫 번째 업그레이드.
 - b. 그런 다음 23.04에서 24.06로 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 [Trident 21.01.1에 대한 발행 세부 정보](#)를 참조하십시오.

2단계: 원래 설치 방법을 결정합니다

Trident를 처음 설치할 때 사용한 버전을 확인하려면 다음을 수행합니다.

1. `tridentctl`을 사용하여 `kubectl get pods -n trident` 포드를 검사합니다.
 - 운영자 포드가 없으면 `tridentctl`을 사용하여 Trident를 설치한 것입니다.
 - 운영자 포드가 있는 경우, Trident는 Trident 운영자를 사용하여 수동으로 또는 Helm을 사용하여 설치되었습니다.
2. 운영자 포드가 있는 경우 `kubectl describe trident`을 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"](#)



작업자와 업그레이드하기 전에 검토할 것을 ["운영자 업그레이드 워크플로우를 이해합니다"](#) 권장합니다.

*

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 DaemonSet의 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 DaemonSet을 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. 운영자는 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 운영자 설치에서 다른 클러스터 범위 Trident 운영자 설치로 업그레이드할 수 있습니다. 모든 Trident 버전 21.01 이상은 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 "지원되는 Kubernetes 버전" 확인합니다.

단계

1. Trident 버전 확인:

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

2. 현재 Trident 인스턴스를 설치하는 데 사용한 Trident 연산자를 삭제합니다. 예를 들어, 23.07에서 업그레이드하는 경우 다음 명령을 실행합니다.

```
kubectl delete -f 23.07.0/trident-installer/deploy/<bundle.yaml> -n trident
```

3. 속성을 사용하여 초기 설치를 사용자 지정한 경우 TridentOrchestrator 개체를 편집하여 설치 매개 변수를 수정할 수 TridentOrchestrator 있습니다. 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 Trident를 bundle_post_1_25.yaml 설치합니다. 여기서 <bundle.yaml>는 bundle_pre_1_25.yaml Kubernetes 버전을 기반으로 합니다. 예를 들어, Trident 24.10을 설치하는 경우 다음 명령을 실행합니다.

```
kubectl create -f 24.10.0/trident-installer/deploy/<bundle.yaml> -n trident
```

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 100.2410.0` 있습니다. Helm repo를 추가하지 않았거나 업그레이드에 사용할 수 없는 경우:
 - a. 에서 최신 Trident 릴리스를 "GitHub의 [_Assets_](#) 섹션" 다운로드하십시오.
 - b. `helm upgrade``에서 업그레이드할 버전을 반영하는 명령을 ``trident-operator-24.10.0.tgz` 사용합니다.

```
helm upgrade <name> trident-operator-24.10.0.tgz
```



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

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

설치에서 **Trident** 운영자로 업그레이드합니다 `tridentctl`

설치 시 Trident 운영자의 최신 릴리스로 업그레이드할 수 `tridentctl` 있습니다. 기존 백엔드 및 PVC를 자동으로 사용할 수 있습니다.



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

단계

1. 최신 Trident 릴리스를 다운로드합니다.

```
# Download the release required [24.10.0]
mkdir 24.10.0
cd 24.10.0
wget
https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

2. ``tridentorchestrator`` 매니페스트에서 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:          v24.10.0
```

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

"[업그레이드 방법을 선택합니다](#)"를 사용하기 전에 tridentctl 검토하십시오.

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

를 실행하여 사용 가능한 명령 목록을 tridentctl 가져오거나 해당 명령에 대한 옵션 및 플래그 목록을 가져오기 위해 해당 명령에 플래그를 추가할 `--help` 수 tridentctl help 있습니다.

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

Trident tridentctl 유틸리티는 다음 명령 및 글로벌 플래그를 지원합니다.

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.

uninstall

Trident를 제거합니다.

update

Trident에서 리소스를 수정합니다.

update backend state

백엔드 작업을 일시적으로 중단합니다.

upgrade

Trident에서 리소스를 업그레이드합니다.

version

Trident 버전을 인쇄합니다.

-d, --debug

디버그 출력.

-h, --help

도움말 `tridentctl`.

-k, --kubeconfig string

'KUBECONFIG' 명령을 로컬로 실행하거나 Kubernetes 클러스터 간에 실행할 경로를 지정합니다.



또는 변수를 내보내어 특정 Kubernetes 클러스터를 가리키도록 하고 해당 클러스터에 명령을 실행할 `tridentctl` 수 KUBECONFIG 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 소스 볼륨으로 쿼리를 제한합니다.
--subordinateOf string: 쿼리를 볼륨의 부하로 제한합니다.

이미지

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

```
tridentctl images [flags]
```

깃발

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

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 파일 경로.
-h, --help: 볼륨에 대한 도움말입니다.
--no-manage: PV/PVC만 생성합니다. 볼륨 라이프사이클 관리를 가정하지 마십시오.

설치합니다

``install`` 플래그를 사용하여 Trident를 설치합니다.

```
tridentctl install [flags]
```

깃발

- autosupport-image string: AutoSupport 원격 측정용 컨테이너 이미지(기본값 "NetApp/Trident AutoSupport:<current-version>").
- autosupport-proxy string: AutoSupport 원격 측정을 보내는 프록시의 주소/포트입니다.
- enable-node-prep: 노드에 필요한 패키지를 설치하려고 시도합니다.
- generate-custom-yaml: 아무것도 설치하지 않고 YAML 파일을 생성합니다.
- h, --help: 설치 도움말.
- http-request-timeout: Trident 컨트롤러의 REST API에 대한 HTTP 요청 시간 초과를 재정의합니다 (기본값 1m30s).
- image-registry string: 내부 이미지 레지스트리의 주소/포트입니다.
- k8s-timeout duration: 모든 Kubernetes 작업에 대한 시간 초과(기본값 3m0s)
- kubelet-dir string: kubelet의 내부 상태의 호스트 위치(기본값 "/var/lib/kubelet").
- log-format string: Trident logging format (text, json) (default "text").
- node-prep: Trident가 지정된 데이터 스토리지 프로토콜을 사용하여 볼륨을 관리할 Kubernetes 클러스터의 노드를 준비할 수 있도록 합니다. 현재 **iscsi** 지원되는 유일한 값입니다.
- pv string: Trident에서 사용하는 레거시 PV의 이름으로 이 이름이 존재하지 않는지 확인합니다(기본값 "Trident").
- pvc string: Trident에서 사용하는 레거시 PVC의 이름으로, 이것이 존재하지 않는지 확인합니다 (기본값 "Trident").
- silence-autosupport: AutoSupport 번들을 NetApp에 자동으로 보내지 않습니다(기본값 true).
- silent: 설치하는 동안 MOST 출력을 비활성화합니다.
- trident-image string: 설치할 Trident 이미지입니다.
- use-custom-yaml: 설치 디렉토리에 있는 기존 YAML 파일을 사용합니다.
- use-ipv6: Trident 통신에 IPv6를 사용합니다.

로그

`logs` 플래그를 사용하여 Trident에서 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 NetApp로 전송합니다.

설치 제거

``uninstall`` 플래그를 사용하여 Trident를 제거합니다.

```
tridentctl uninstall [flags]
```

깃발

- `-h, --help`: 제거에 대한 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

명령을 사용하여 update backend state 백엔드 작업을 일시 중단하거나 재개합니다.

```
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`: 백엔드 작업을 일시 중지하려면 으로 suspended 설정합니다. 백엔드 작업을 재개하려면 으로 normal 설정합니다. 다음으로 설정된 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" "kubect를 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. 업데이트된 파일의 경로와 명령을 사용하여 백엔드를 `tridentctl backend update backend.json` 업데이트합니다.

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

```
{
  "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

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

```
tridentctl version [flags]
```

깃발

- `--client`: 클라이언트 버전만 (서버가 필요하지 않음).
- `-h, --help`: 버전 도움말.

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, tridentctl은 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 `/metrics` 끝점의 대상 포트에 노출됩니다. 8001 Trident가 설치된 경우 이러한 메트릭은 기본적으로 * 활성화됩니다.

필요한 것

- Trident가 설치된 Kubernetes 클러스터
- 프로메테우스(Prometheus) 인스턴스. "[컨테이너형 Prometheus 구축](#)" 또는 Prometheus를 로 실행하도록 선택할 수 "[네이티브 애플리케이션](#)" 있습니다.

1단계: Prometheus 목표를 정의합니다

Prometheus 타겟을 정의하여 메트릭을 수집하고 Trident가 관리하는 백엔드, 생성된 볼륨 등에 대한 정보를 얻어야

합니다. ["블로그"](#)Trident와 함께 Prometheus 및 Grafana를 사용하여 메트릭을 검색하는 방법에 대해 설명합니다. 블로그에서 Prometheus를 Kubernetes 클러스터의 운영자로 실행하고 ServiceMonitor를 생성하여 Trident 메트릭을 얻는 방법을 설명합니다.

2단계: Prometheus ServiceMonitor를 만듭니다

Trident 메트릭을 사용하려면 서비스를 감시하고 포트에서 수신 대기하는 metrics Prometheus ServiceMonitor 를 만들어야 trident-csi 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 서비스에서 metrics 반환된 메트릭을 검색하여 trident-csi 서비스의 끝점을 찾습니다. 그 결과, Prometheus는 이제 Trident의 메트릭을 이해할 수 있도록 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.

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

- **Trident의 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 Trident와 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. 이는 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:
    disableDelete: false
    serialNumbers:
    - nwkvzfanek_SN
    limitVolumeSize: ''
  state: online
  online: true
```

- AutoSupport 메시지는 NetApp의 AutoSupport 엔드포인트로 전송됩니다. 개인 레지스트리를 사용하여 컨테이너 이미지를 저장하는 경우 플래그를 사용할 수 `--image-registry` 있습니다.
- 또한 설치 YAML 파일을 생성하여 프록시 URL을 구성할 수도 있습니다. 을 사용하여 YAML 파일을 만들고 `--proxy-url` 에서 컨테이너에 `trident-deployment.yaml` 대한 인수를 `trident-autosupport` 추가하면 이 작업을 수행할 수 `tridentctl install --generate-custom-yaml` 있습니다.

Trident 메트릭을 비활성화합니다

****메트릭이 보고되지 않도록 하려면 플래그를 사용하여 사용자 지정 YAML을 생성하고 편집하여 `--metrics` 컨테이너에 대한 플래그가 호출되지 않도록 `trident-main` 해야 `--generate-custom-yaml` 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 ["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.

- 를 사용하여 `kubectl get pods -n trident` 포드를 검사합니다.
 - 운영자 포드가 없으면 를 사용하여 Trident를 설치한 `tridentctl` 것입니다.
 - 운영자 포드가 있는 경우, Trident는 Trident 운영자를 사용하여 수동으로 또는 Helm을 사용하여 설치되었습니다.
- 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.

- CR 편집 `TridentOrchestrator` 및 제거 플래그 설정:**

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

플래그가 로 설정된 `true` 경우 `uninstall` Trident 운영자는 Trident를 제거하지만 `TridentOrchestrator` 자체는 제거하지 않습니다. Trident를 다시 설치하려면 해당 Trident를 정리하고 새 `AgentOrchestrator`를 생성해야 합니다.

- 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
```

설치를 제거합니다 tridentctl

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

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

저작권 정보

Copyright © 2025 NetApp, Inc. All Rights Reserved. 미국에서 인쇄된 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 라이선스와 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 라이선스도 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 라이선스를 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 라이선스는 DFARS 조항 252.227-7015(b)(2014년 2월)에 명시된 권한으로 제한됩니다.

상표 정보

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