스토리지 클래스 생성 및 관리

Trident

NetApp
March 05, 2026

목차

스토리지 클래스 생성 및 관리 . . . 1
- 스토리지 클래스 생성 . . . 1
  - Kubernetes StorageClass 객체 구성 . . . 1
  - 스토리지 클래스 생성 . . . 1
- 스토리지 클래스 관리 . . . 3
  - 기존 스토리지 클래스 보기 . . . 3
  - 기본 저장소 클래스 설정 . . . 4
  - 스토리지 클래스의 백엔드 식별 . . . 4
  - 스토리지 클래스 삭제 . . . 4

스토리지 클래스 생성 및 관리

스토리지 클래스 생성

Kubernetes StorageClass 객체를 구성하고 Trident 가 볼륨을 프로비저닝하는 방법을 지시하는 스토리지 클래스를 만듭니다.

Kubernetes StorageClass 객체 구성

그만큼 "Kubernetes StorageClass 객체" Trident 해당 클래스에 사용되는 프로비저너로 식별하고 Trident 볼륨을 프로비저닝하는 방법을 지시합니다. 예를 들어:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
mountOptions:
  - nfsvers=3
  - nolock
parameters:
  backendType: "ontap-nas"
  media: "ssd"
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

참조하다"Kubernetes 및 Trident 객체" 저장소 클래스가 어떻게 상호 작용하는지에 대한 자세한 내용은 다음과 같습니다. `PersistentVolumeClaim` Trident 가 볼륨을 프로비저닝하는 방식을 제어하기 위한 매개변수입니다.

스토리지 클래스 생성

StorageClass 객체를 만든 후에는 스토리지 클래스를 만들 수 있습니다. 저장 클래스 샘플 사용하거나 수정할 수 있는 몇 가지 기본 샘플을 제공합니다.

단계

1. 이것은 Kubernetes 객체이므로 다음을 사용합니다. `kubectl` Kubernetes에서 생성하세요.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

2. 이제 Kubernetes와 Trident 모두에서 **basic-csi** 스토리지 클래스를 볼 수 있어야 하며, Trident 가 백엔드에서 풀을 검색했어야 합니다.

```
kubectl get sc basic-csi
```

```
NAME          PROVISIONER             AGE
basic-csi     csi.trident.netapp.io   15h
```

```
./tridentctl -n trident get storageclass basic-csi -o json
```

```
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

저장 클래스 샘플

Trident 제공합니다 "특정 백엔드에 대한 간단한 스토리지 클래스 정의" .

또는 편집할 수 있습니다 `sample-input/storage-class-csi.yaml.templ` 설치 프로그램과 함께 제공되는 파일을 교체합니다. *`BACKEND_TYPE`* 저장소 드라이버 이름으로.

```
./tridentctl -n trident get backend
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

스토리지 클래스 관리

기존 스토리지 클래스를 보고, 기본 스토리지 클래스를 설정하고, 스토리지 클래스 백엔드를 식별하고, 스토리지 클래스를 삭제할 수 있습니다.

기존 스토리지 클래스 보기

- 기존 Kubernetes 스토리지 클래스를 보려면 다음 명령을 실행하세요.

```
kubectl get storageclass
```

- Kubernetes 스토리지 클래스 세부 정보를 보려면 다음 명령을 실행하세요.

```
kubectl get storageclass <storage-class> -o json
```

- Trident의 동기화된 스토리지 클래스를 보려면 다음 명령을 실행하세요.

```
tridentctl get storageclass
```

- Trident의 동기화된 스토리지 클래스 세부 정보를 보려면 다음 명령을 실행하세요.

```
tridentctl get storageclass <storage-class> -o json
```

기본 저장소 클래스 설정

Kubernetes 1.6에서는 기본 스토리지 클래스를 설정하는 기능이 추가되었습니다. 이는 사용자가 PVC(영구 볼륨 클레임)에서 영구 볼륨을 지정하지 않은 경우 영구 볼륨을 프로비저닝하는 데 사용되는 스토리지 클래스입니다.

- 주석을 설정하여 기본 저장 클래스를 정의합니다. `storageclass.kubernetes.io/is-default-class` 저장 클래스 정의에서 true로 설정합니다. 사양에 따르면, 다른 값이나 주석이 없는 경우 거짓으로 해석됩니다.
- 다음 명령을 사용하여 기존 스토리지 클래스를 기본 스토리지 클래스로 구성할 수 있습니다.

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- 마찬가지로 다음 명령을 사용하여 기본 저장소 클래스 주석을 제거할 수 있습니다.

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Trident 설치 프로그램 번들에는 이 주석이 포함된 예도 있습니다.

> 클러스터에는 한 번에 하나의 기본 스토리지 클래스만 있어야 합니다. Kubernetes는 기술적으로 두 개 이상을 갖는 것을 막지는 않지만, 기본 스토리지 클래스가 전혀 없는 것처럼 동작합니다.

스토리지 클래스의 백엔드 식별

이는 JSON으로 답할 수 있는 질문 종류의 예입니다. `tridentctl` Trident 백엔드 객체에 대한 출력. 이것은 다음을 사용합니다 `jq` 먼저 설치해야 할 유틸리티가 있습니다.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

스토리지 클래스 삭제

Kubernetes에서 스토리지 클래스를 삭제하려면 다음 명령을 실행하세요.

```
kubectl delete storageclass <storage-class>
```

`<storage-class>`저장 클래스로 대체해야 합니다.

이 스토리지 클래스를 통해 생성된 모든 영구 볼륨은 변경되지 않으며 Trident 계속해서 이를 관리합니다.

Trident 공백을 강화합니다. `fsType` 그것이 만들어내는 볼륨 때문에요. iSCSI 백엔드의 경우 다음을 적용하는 것이 좋습니다. `parameters.fsType` StorageClass에서. 기존 StorageClass를 삭제하고 다시 생성해야 합니다. `parameters.fsType` 지정된.