백엔드 관리

Trident

NetApp
July 01, 2026

목차

백엔드 관리

kubectl을 사용하여 백엔드 관리를 수행합니다

`kubectl`를 사용하여 백엔드 관리 작업을 수행하는 방법에 대해 알아보십시오.

백엔드를 삭제합니다

`TridentBackendConfig`를 삭제하면 Trident에 백엔드를 삭제/유지하도록(`deletionPolicy`에 따라) 지시하는 것입니다. 백엔드를 삭제하려면 `deletionPolicy`가 delete로 설정되어 있는지 확인하십시오. `TridentBackendConfig`만 삭제하려면 `deletionPolicy`가 retain으로 설정되어 있는지 확인하십시오. 이렇게 하면 백엔드가 계속 존재하며 `tridentctl`를 사용하여 관리할 수 있습니다.

다음 명령을 실행합니다.

```
kubectl delete tbc <tbc-name> -n trident
```

Trident는 `TridentBackendConfig`에서 사용 중인 Kubernetes Secret을 삭제하지 않습니다. Kubernetes 사용자가 Secret 정리를 담당합니다. Secret을 삭제할 때는 주의해야 합니다. 백엔드에서 사용하지 않는 Secret만 삭제해야 합니다.

기존 백엔드 보기

다음 명령을 실행합니다.

```
kubectl get tbc -n trident
```

`tridentctl get backend -n trident` 또는 `tridentctl get backend -o yaml -n trident`를 실행하여 존재하는 모든 백엔드 목록을 얻을 수도 있습니다. 이 목록에는 `tridentctl`로 생성된 백엔드도 포함됩니다.

백엔드 업데이트

백엔드를 업데이트해야 하는 이유는 여러 가지가 있을 수 있습니다.

- 스토리지 시스템에 대한 자격 증명이 변경되었습니다. 자격 증명을 업데이트하려면 TridentBackendConfig 객체에 사용되는 Kubernetes Secret을 업데이트해야 합니다. Trident는 제공된 최신 자격 증명으로 백엔드를 자동으로 업데이트합니다. 다음 명령을 실행하여 Kubernetes Secret을 업데이트하십시오.

```
kubectl apply -f <updated-secret-file.yaml> -n trident
```

- 매개변수(예: 사용 중인 ONTAP SVM의 이름)를 업데이트해야 합니다.
  - 다음 명령을 사용하여 Kubernetes를 통해 `TridentBackendConfig` 객체를 직접 업데이트할 수 있습니다.

```
kubectl apply -f <updated-backend-file.yaml>
```

  - 또는 다음 명령을 사용하여 기존 `TridentBackendConfig` CR을 변경할 수 있습니다.

```
kubectl edit tbc <tbc-name> -n trident
```

- 백엔드 업데이트가 실패하면 백엔드는 마지막으로 알려진 구성을 계속 유지합니다. 로그를 확인하여 원인을 파악하려면 `kubectl get tbc <tbc-name> -o yaml -n trident` 또는 `kubectl describe tbc <tbc-name> -n trident`을 실행하십시오.
- 구성 파일의 문제를 식별하고 수정한 후 업데이트 명령을 다시 실행할 수 있습니다.

tridentctl을 사용하여 백엔드 관리를 수행합니다

`tridentctl`를 사용하여 백엔드 관리 작업을 수행하는 방법에 대해 알아보십시오.

백엔드 생성

"백엔드 구성 파일"를 생성한 후 다음 명령을 실행하십시오.

```
tridentctl create backend -f <backend-file> -n trident
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하여 로그를 확인하고 원인을 파악할 수 있습니다.

```
tridentctl logs -n trident
```

구성 파일의 문제를 식별하고 수정한 후 `create` 명령을 다시 실행하기만 하면 됩니다.

백엔드를 삭제합니다

Trident에서 백엔드를 삭제하려면 다음 단계를 따르세요.

1. 백엔드 이름 검색:

```
tridentctl get backend -n trident
```

2. 백엔드 삭제:

```
tridentctl delete backend <backend-name> -n trident
```

Trident가 이 백엔드에서 프로비저닝한 볼륨과 스냅샷이 아직 남아 있는 경우, 백엔드를 삭제하면 해당 백엔드에서 새 볼륨을 프로비저닝할 수 없습니다. 백엔드는 "Deleting" 상태로 계속 유지됩니다.

기존 백엔드 보기

Trident가 알고 있는 백엔드를 보려면 다음을 수행합니다.

- 요약을 보려면 다음 명령을 실행하십시오.

```
tridentctl get backend -n trident
```

- 모든 세부 정보를 확인하려면 다음 명령을 실행하십시오.

```
tridentctl get backend -o json -n trident
```

백엔드 업데이트

새 백엔드 구성 파일을 생성한 후 다음 명령을 실행하십시오.

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

백엔드 업데이트가 실패하면 백엔드 구성에 문제가 있거나 유효하지 않은 업데이트를 시도한 것입니다. 다음 명령을 실행하여 로그를 확인하고 원인을 파악할 수 있습니다.

```
tridentctl logs -n trident
```

구성 파일의 문제를 식별하고 수정한 후 `update` 명령을 다시 실행하기만 하면 됩니다.

백엔드를 사용하는 스토리지 클래스를 식별합니다

이는 백엔드 객체에 대해 `tridentctl`이(가) 출력하는 JSON을 사용하여 답변할 수 있는 질문 유형의 예입니다. 이는 설치해야 하는 `jq` 유틸리티를 사용합니다.

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

이는 `TridentBackendConfig`을 사용하여 생성된 백엔드에도 적용됩니다.

백엔드 관리 옵션 간 이동

Trident에서 백엔드를 관리하는 다양한 방법에 대해 알아보십시오.

백엔드 관리 옵션

`TridentBackendConfig`의 도입으로 관리자는 이제 백엔드를 관리하는 두 가지 고유한 방법을 갖게 되었습니다. 이로 인해 다음과 같은 질문이 제기됩니다.

- `tridentctl`를 사용하여 생성된 백엔드를 `TridentBackendConfig`로 관리할 수 있습니까?
- `TridentBackendConfig`를 사용하여 생성된 백엔드를 `tridentctl`를 사용하여 관리할 수 있습니까?

`tridentctl`를 사용하여 `TridentBackendConfig` 백엔드 관리

이 섹션에서는 Kubernetes 인터페이스를 통해 직접 `tridentctl`를 사용하여 생성된 백엔드를 관리하기 위해 `TridentBackendConfig` 오브젝트를 생성하는 데 필요한 단계를 다룹니다.

다음 시나리오에 적용됩니다.

- 기존 백엔드의 경우, `TridentBackendConfig`가 없으며, 이는 `tridentctl`로 생성되었기 때문입니다.
- `tridentctl`로 생성된 새 백엔드, 다른 `TridentBackendConfig` 객체가 존재합니다.

두 시나리오 모두에서 백엔드는 계속 존재하며 Trident는 볼륨을 예약하고 운영합니다. 관리자는 여기에서 두 가지 중 하나를 선택할 수 있습니다.

- `tridentctl`을 사용하여 생성한 백엔드를 계속 관리합니다.
- 생성된 백엔드를 `tridentctl` 새 `TridentBackendConfig` 오브젝트에 바인딩합니다. 이렇게 하면 백엔드는 `kubectl`를 사용하여 관리되고 `tridentctl`로는 관리되지 않습니다.

  `kubectl`을 사용하여 기존 백엔드를 관리하려면 기존 백엔드에 바인딩하는 `TridentBackendConfig`를 만들어야 합니다. 다음은 그 작동 방식에 대한 개요입니다.

  1. Kubernetes Secret을 생성합니다. 이 시크릿에는 Trident가 스토리지 클러스터/서비스와 통신하는 데 필요한 자격 증명이 포함되어 있습니다.
  2. `TridentBackendConfig` 개체를 만듭니다. 여기에는 스토리지 클러스터/서비스에 대한 세부 정보가 포함되어 있으며 이전 단계에서 만든 시크릿을 참조합니다. 동일한 구성 매개변수(예: `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName` 등)를 지정하도록 주의해야 합니다. `spec.backendName`는 기존 백엔드의 이름으로 설정해야 합니다.

0단계: 백엔드 식별

기존 백엔드에 바인딩하는 `TridentBackendConfig`를 만들려면 백엔드 구성을 가져와야 합니다. 이 예에서는 다음 JSON 정의를 사용하여 백엔드를 만들었다고 가정합니다.

```
tridentctl get backend ontap-nas-backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

```
cat ontap-nas-backend.json
```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.10.10.1",
  "dataLIF": "10.10.10.2",
  "backendName": "ontap-nas-backend",
  "svm": "trident_svm",
  "username": "cluster-admin",
  "password": "admin-password",
  "defaults": {
    "spaceReserve": "none",
    "encryption": "false"
  },
  "labels": {
    "store": "nas_store"
  },
  "region": "us_east_1",
  "storage": [
    {
      "labels": {
        "app": "msoffice",
        "cost": "100"
      },
      "zone": "us_east_1a",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "true",
        "unixPermissions": "0755"
      }
    },
    {
      "labels": {
        "app": "mysqldb",
        "cost": "25"
      },
      "zone": "us_east_1d",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "false",
        "unixPermissions": "0775"
      }
    }
  ]
}
```

1단계: **Kubernetes Secret** 생성

이 예에 표시된 대로 백엔드에 대한 자격 증명이 포함된 Secret을 생성합니다.

```
cat tbc-ontap-nas-backend-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  password: admin-password
```

```
kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

2단계: `TridentBackendConfig` **CR** 생성

다음 단계는 기존에 존재하는 `ontap-nas-backend`에 자동으로 바인딩되는 `TridentBackendConfig` CR을 생성하는 것입니다(이 예시와 같이). 다음 요구 사항이 충족되는지 확인하십시오:

- 동일한 백엔드 이름이 `spec.backendName`에 정의되어 있습니다.
- 구성 매개변수는 원래 백엔드와 동일합니다.
- 가상 풀(있는 경우)은 원래 백엔드에서와 동일한 순서를 유지해야 합니다.
- 자격 증명은 일반 텍스트가 아닌 Kubernetes Secret을 통해 제공됩니다.

이 경우 `TridentBackendConfig`는 다음과 같이 표시됩니다:

```
cat backend-tbc-ontap-nas.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
      app: msoffice
      cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
      app: mysqldb
      cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'
```

```
kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

3단계: `TridentBackendConfig` **CR**의 상태를 확인합니다

`TridentBackendConfig`을 생성한 후 해당 단계는 `Bound`이어야 합니다. 또한 기존 백엔드의 이름 및 UUID와 동일한 백엔드 이름을 반영해야 합니다.

```
kubectl get tbc tbc-ontap-nas-backend -n trident
NAME                    BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                  UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

이제 백엔드는 tbc-ontap-nas-backend TridentBackendConfig 객체를 사용하여 완전히 관리됩니다.

TridentBackendConfig`를 사용하여 `tridentctl 백엔드 관리

`tridentctl`는 `TridentBackendConfig`를 사용하여 생성된 백엔드를 나열하는 데 사용할 수 있습니다. 또한, 관리자는 `tridentctl`를 통해 이러한 백엔드를 완전히 관리하도록 선택할 수 있으며, `TridentBackendConfig`를 삭제하고 `spec.deletionPolicy`가 `retain`로 설정되어 있는지 확인할 수 있습니다.

0단계**:** 백엔드 식별

예를 들어 다음 백엔드가 `TridentBackendConfig`을 사용하여 생성되었다고 가정해 보겠습니다:

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

출력에서 `TridentBackendConfig`이(가) 성공적으로 생성되었으며 백엔드에 바인딩된 것을 볼 수 있습니다[백엔드의 UUID 관찰].

1단계: **`deletionPolicy`**가 **`retain`**로 설정되어 있는지 확인합니다

`deletionPolicy`의 값을 살펴보겠습니다. 이것은 `retain`로 설정해야 합니다. 이렇게 하면 `TridentBackendConfig` CR이 삭제되더라도 백엔드 정의는 계속 존재하며 `tridentctl`로 관리할 수 있습니다.

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

ⓘ `deletionPolicy`이(가) `retain`(으)로 설정되지 않은 경우 다음 단계로 진행하지 마십시오.

2단계: `TridentBackendConfig` **CR 삭제**

마지막 단계는 `TridentBackendConfig` CR을 삭제하는 것입니다. `deletionPolicy`가 `retain`로 설정되어 있는지 확인한 후 삭제를 진행하면 됩니다:

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

`TridentBackendConfig` 개체를 삭제하면 `Trident`는 백엔드 자체를 실제로 삭제하지 않고 단순히 제거합니다.

저작권 정보

상표 정보