管理后端

Trident

NetApp
January 15, 2026

目录

管理后端 . . . 1
使用 kubectl 执行后端管理 . . . 1
删除后端 . . . 1
查看现有后端 . . . 1
更新后端 . . . 1
使用 tridentctl 执行后端管理 . . . 2
创建后端 . . . 2
删除后端 . . . 2
查看现有后端 . . . 3
更新后端 . . . 3
确定使用后端存储的存储类 . . . 3
在后端管理选项之间切换 . . . 3
后端管理选项 . . . 4
管理 `tridentctl` 使用后端 `TridentBackendConfig` . . . 4
管理 `TridentBackendConfig` 使用后端 `tridentctl` . . . 9

管理后端

使用 kubectl 执行后端管理

了解如何使用以下方式执行后端管理操作 kubectl。

删除后端

通过删除一个 TridentBackendConfig`您指示Trident删除/保留后端（基于 `deletionPolicy）。要删除后端，请确保 deletionPolicy`已设置为删除。仅删除 `TridentBackendConfig`确保 `deletionPolicy`设置为保留。这样可以确保后端仍然存在，并且可以通过以下方式进行管理：`tridentctl 。

运行以下命令：

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

Trident不会删除正在使用的 Kubernetes Secrets。 TridentBackendConfig 。 Kubernetes 用户负责清理密钥。删除机密信息时务必谨慎。只有当后端不再使用密钥时，才应该删除密钥。

查看现有后端

运行以下命令：

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

你也可以运行 tridentctl get backend -n trident`或者 `tridentctl get backend -o yaml -n trident`获取所有现有后端的列表。此列表还将包括使用以下方式创建的后端： `tridentctl 。

更新后端

更新后端的原因可能有很多：

- 存储系统的凭证已更改。要更新凭据，需要更新 Kubernetes Secret，该 Secret 用于：`TridentBackendConfig`对象必须更新。 Trident会自动使用提供的最新凭据更新后端。运行以下命令更新 Kubernetes Secret：

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

- 需要更新参数（例如正在使用的ONTAP SVM 的名称）。
  - 您可以更新 `TridentBackendConfig`使用以下命令直接通过 Kubernetes 访问对象：

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

- 或者，您可以对现有内容进行更改。 `TridentBackendConfig`使用以下命令进行回车：

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

- 如果后端更新失败，后端将继续保持其最后一次已知的配置。您可以通过运行以下命令查看日志以确定原因。 `kubectl get tbc <tbc-name> -o yaml -n trident`或者 `kubectl describe tbc <tbc-name> -n trident`。
- 在您发现并纠正配置文件中的问题后，您可以重新运行更新命令。

使用 tridentctl 执行后端管理

了解如何使用以下方式执行后端管理操作 `tridentctl`。

创建后端

创建之后"后端配置文件"运行以下命令：

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

如果后端创建失败，则说明后端配置存在问题。您可以通过运行以下命令查看日志以确定原因：

```
tridentctl logs -n trident
```

在您发现并纠正配置文件中的问题后，您只需运行以下命令即可。 `create`再次发出命令。

删除后端

要从Trident中删除后端，请执行以下操作：

1. 获取后端名称：

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

2. 删除后端：

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

如果Trident已从此后端配置了仍然存在的卷和快照，则删除后端将阻止从该后端配置新卷。后端将继续处于“删除”状态。

查看现有后端

要查看Trident已知的后端，请执行以下操作：

- 要获取摘要，请运行以下命令：

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

- 要获取所有详细信息，请运行以下命令：

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

更新后端

创建新的后端配置文件后，运行以下命令：

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

如果后端更新失败，则说明后端配置有问题，或者您尝试了无效的更新。您可以通过运行以下命令查看日志以确定原因：

```
tridentctl logs -n trident
```

在您发现并纠正配置文件中的问题后，您只需运行以下命令即可。 `update`再次发出命令。

确定使用后端存储的存储类

这是一个您可以使用 JSON 回答的问题示例： `tridentctl`后端对象的输出。这使用 `jq`您需要安装该实用程序。

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

这也适用于使用以下方式创建的后端： `TridentBackendConfig` 。

在后端管理选项之间切换

了解Trident中管理后端的不同方法。

后端管理选项

随着 `TridentBackendConfig`现在，管理员有两种独特的后端管理方式。这就引出了以下问题：

- 可以使用以下方式创建后端 `tridentctl`以……进行管理 `TridentBackendConfig`?
- 可以使用以下方式创建后端 `TridentBackendConfig`可通过以下方式进行管理 `tridentctl`?

管理 `tridentctl`使用后端 `TridentBackendConfig`

本节介绍管理使用以下方式创建的后端所需的步骤： `tridentctl`直接通过 Kubernetes 接口创建 `TridentBackendConfig`物体。

这适用于以下情况：

- 预先存在的后端，没有 `TridentBackendConfig`因为它们是用……创造的 `tridentctl`。
- 使用以下方式创建的新后端 `tridentctl`而其他 `TridentBackendConfig`物体是存在的。

在这两种情况下，后端都将继续存在， Trident将调度卷并对其进行操作。管理员此时有两种选择：

- 继续使用 `tridentctl`管理使用它创建的后端。
- 使用以下方式创建的绑定后端 `tridentctl`到一个新的 `TridentBackendConfig`目的。这样做意味着后端将使用以下方式进行管理： `kubectl`而不是 `tridentctl`。

使用以下方式管理预先存在的后端 `kubectl`您需要创建一个 `TridentBackendConfig`它与现有后端绑定。以下是其工作原理概述：

1. 创建 Kubernetes Secret。该密钥包含Trident与存储集群/服务通信所需的凭据。
2. 创建一个 `TridentBackendConfig`目的。这包含有关存储集群/服务的具体信息，并引用上一步中创建的密钥。必须注意指定完全相同的配置参数（例如： `spec.backendName`， `spec.storagePrefix`， `spec.storageDriverName`， 等等）。 `spec.backendName`必须设置为现有后端的名称。

步骤 0：确定后端

创建一个 `TridentBackendConfig`如果要绑定到现有后端，则需要获取后端配置。在这个例子中，我们假设使用以下 JSON 定义创建了一个后端：

```
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"
      }
    }
  ]
}
```

步骤 1：创建 **Kubernetes Secret**

创建一个包含后端凭据的 Secret，如下例所示：

```
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
```

步骤 2：创建 **`TridentBackendConfig`CR**

下一步是创建一个 `TridentBackendConfig`CR` 将自动绑定到预先存在的 `ontap-nas-backend（如本例所示）。请确保满足以下要求：

- 后端名称在以下位置定义： `spec.backendName` 。
- 配置参数与原后端相同。
- 虚拟池（如果存在）必须保持与原始后端相同的顺序。
- 凭证通过 Kubernetes Secret 提供，而不是以明文形式提供。

在这种情况下， `TridentBackendConfig`将会像这样：

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

```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
```

步骤 3：验证状态 `TridentBackendConfig`CR

之后 `TridentBackendConfig`已经创建，它的阶段必须是 `Bound`。它还应该反映与现有后端相同的后端名称和 UUID。

```
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 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

后端现在将完全使用以下方式进行管理： `tbc-ontap-nas-backend` `TridentBackendConfig`目的。

管理 `TridentBackendConfig`使用后端 `tridentctl`

`tridentctl`可用于列出使用以下方式创建的后端： `TridentBackendConfig` 。此外，管理员还可以选择通过以下方式完全管理此类后端： `tridentctl`通过删除 `TridentBackendConfig`并确保 `spec.deletionPolicy`设置为 `retain`。

步骤 **0**：确定后端

例如，假设我们使用以下方式创建了以下后端 `TridentBackendConfig`：

```
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 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

从输出结果可以看出： `TridentBackendConfig`已成功创建并绑定到后端[观察后端的 UUID]。

步骤 **1**：确认 `deletionPolicy`设置为 `retain`

让我们来看看它的价值 `deletionPolicy`。需要将其设置为 `retain`。这确保了当 `TridentBackendConfig`CR 被删除后，后端定义仍然存在，并且可以通过以下方式进行管理： `tridentctl`。

```
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
```

除非另有说明，否则请勿进行下一步。 `deletionPolicy`设置为 `retain。

步骤二：删除 `TridentBackendConfig`CR

最后一步是删除 `TridentBackendConfig`CR。确认后 `deletionPolicy`设置为 `retain`您可以继续删除：

```
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 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

删除后 `TridentBackendConfig`Trident只是移除该对象，而不会实际删除后端本身。

版权信息

商标信息