Gestiona backends

Trident

NetApp
July 01, 2026

Tabla de contenidos

Gestiona backends

Realiza la gestión del backend con kubectl

Conoce cómo realizar operaciones de gestión de backend usando `kubectl`.

Eliminar un backend

Al eliminar un `TridentBackendConfig`, le indicas a Trident que elimine o conserve los backends (según `deletionPolicy`). Para eliminar un backend, asegúrate de que `deletionPolicy` esté configurado para eliminar. Para eliminar solo el `TridentBackendConfig`, asegúrate de que `deletionPolicy` esté configurado para conservar. Esto asegura que el backend siga presente y se pueda administrar usando `tridentctl`.

Ejecuta el siguiente comando:

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

Trident no elimina los secretos de Kubernetes que estaban en uso por `TridentBackendConfig`. El usuario de Kubernetes es responsable de limpiar los secretos. Debes tener cuidado al eliminar secretos. Solo deberías eliminar secretos si no están en uso por los backends.

Ver los backends existentes

Ejecuta el siguiente comando:

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

También puedes ejecutar `tridentctl get backend -n trident` o `tridentctl get backend -o yaml -n trident` para obtener una lista de todos los backends que existen. Esta lista también incluirá los backends que fueron creados con `tridentctl`.

Actualizar un backend

Puede haber varias razones para actualizar un backend:

- Las credenciales del sistema de almacenamiento han cambiado. Para actualizar las credenciales, se debe actualizar el secreto de Kubernetes que se usa en el objeto `TridentBackendConfig`. Trident actualizará automáticamente el backend con las credenciales más recientes proporcionadas. Ejecuta el siguiente comando para actualizar el secreto de Kubernetes:

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

- Es necesario actualizar los parámetros (como el nombre del ONTAP SVM que se está utilizando).
  - Puedes actualizar `TridentBackendConfig` objetos directamente a través de Kubernetes usando el siguiente comando:

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

- Alternativamente, puedes hacer cambios en el `TridentBackendConfig`CR existente usando el siguiente comando:

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

- Si falla una actualización del backend, este sigue en su última configuración conocida. Puedes ver los registros para determinar la causa ejecutando `kubectl get tbc <tbc-name> -o yaml -n trident` o `kubectl describe tbc <tbc-name> -n trident`.
- Después de identificar y corregir el problema con el archivo de configuración, puedes volver a ejecutar el comando de actualización.

Realiza la gestión del backend con tridentctl

Conoce cómo realizar operaciones de gestión de backend usando `tridentctl`.

Crear un backend

Después de crear un "archivo de configuración backend", ejecuta el siguiente comando:

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

Si falla la creación del backend, algo estaba mal con la configuración del backend. Puedes ver los registros para determinar la causa ejecutando el siguiente comando:

```
tridentctl logs -n trident
```

Después de identificar y corregir el problema con el archivo de configuración, simplemente puedes ejecutar el comando `create` otra vez.

Eliminar un backend

Para eliminar un backend de Trident, haz lo siguiente:

1. Recupera el nombre del backend:

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

2. Elimina el backend:

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

Si Trident ha aprovisionado volúmenes e instantáneas de este backend que aún existen, eliminar el backend impide que se aprovisionen nuevos volúmenes desde él. El backend seguirá existiendo en estado "Eliminando".

Ver los backends existentes

Para ver los backends que Trident conoce, haz lo siguiente:

- Para obtener un resumen, ejecuta el siguiente comando:

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

- Para obtener todos los detalles, ejecuta el siguiente comando:

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

Actualizar un backend

Después de crear un nuevo archivo de configuración de backend, ejecuta el siguiente comando:

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

Si la actualización del backend falla, algo salió mal con la configuración del backend o intentaste una actualización no válida. Puedes ver los registros para determinar la causa ejecutando el siguiente comando:

```
tridentctl logs -n trident
```

Después de identificar y corregir el problema con el archivo de configuración, simplemente puedes ejecutar el comando `update` otra vez.

Identifica las clases de almacenamiento que usan un backend

Este es un ejemplo del tipo de preguntas que puedes responder con el JSON que `tridentctl` genera para los objetos backend. Esto usa la utilidad `jq`, que necesitas instalar.

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

Esto también se aplica a los backends que se crearon usando `TridentBackendConfig`.

Moverte entre las opciones de gestión del backend

Conoce las diferentes formas de administrar backends en Trident.

Opciones para gestionar backends

Con la introducción de `TridentBackendConfig`, los administradores ahora tienen dos formas únicas de gestionar los backends. Esto plantea las siguientes preguntas:

- ¿Se pueden gestionar los backends creados usando `tridentctl` con `TridentBackendConfig`?
- ¿Se pueden gestionar los backends creados usando `TridentBackendConfig` mediante `tridentctl`?

Administra `tridentctl`backends usando `TridentBackendConfig`

Esta sección cubre los pasos necesarios para administrar backends que se crearon usando `tridentctl` directamente a través de la interfaz de Kubernetes creando `TridentBackendConfig` objetos.

Esto se aplicará a los siguientes escenarios:

- Backends preexistentes que no tienen un `TridentBackendConfig` porque fueron creados con `tridentctl`.
- Nuevos backends que se crearon con `tridentctl`, mientras que existen otros `TridentBackendConfig` objetos.

En ambos escenarios, los backends seguirán presentes, con Trident programando los volúmenes y operando sobre ellos. Aquí los administradores tienen dos opciones:

- Sigue usando `tridentctl` para administrar los backends que se crearon con él.
- Vincula los backends creados usando `tridentctl` un nuevo `TridentBackendConfig` objeto. Hacer esto significa que los backends se gestionarán usando `kubectl` y no `tridentctl`.

Para administrar un backend preexistente usando `kubectl`, necesitarás crear un `TridentBackendConfig` que se vincule al backend existente. Aquí tienes un resumen de cómo funciona:

1. Crea un secreto de Kubernetes. El secreto contiene las credenciales que Trident necesita para comunicarse con el clúster/servicio de almacenamiento.
2. Crea un `TridentBackendConfig` objeto. Esto contiene detalles sobre el clúster o servicio de almacenamiento y hace referencia al secreto creado en el paso anterior. Debes asegurarte de especificar parámetros de configuración idénticos (como `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`, y así sucesivamente). `spec.backendName` debe establecerse con el nombre del backend existente.

Paso 0: identifica el backend

Para crear un `TridentBackendConfig` que se vincule a un backend existente, necesitas obtener la configuración del backend. En este ejemplo, supongamos que se creó un backend usando la siguiente definición JSON:

```
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"
      }
    }
  ]
}
```

Paso 1: crea un secreto de Kubernetes

Crea un secreto que contenga las credenciales para el backend, como se muestra en este ejemplo:

```
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
```

Paso 2: crea un `TridentBackendConfig`CR

El siguiente paso es crear una `TridentBackendConfig` CR que se vincule automáticamente con la preexistente `ontap-nas-backend` (como en este ejemplo). Asegúrate de que se cumplan los siguientes requisitos:

- El mismo nombre de backend se define en `spec.backendName`.
- Los parámetros de configuración son idénticos al backend original.
- Los pools virtuales (si están presentes) deben conservar el mismo orden que en el backend original.
- Las credenciales se proporcionan a través de un Kubernetes Secret y no en texto sin formato.

En este caso, el `TridentBackendConfig` se verá así:

```
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
```

Paso 3: verifica el estado de la `TridentBackendConfig`CR

Después de que el `TridentBackendConfig` haya sido creado, su fase debe ser `Bound`. También debe reflejar el mismo nombre de backend y UUID que el backend existente.

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

Ahora el backend se va a administrar completamente usando el `tbc-ontap-nas-backend TridentBackendConfig` objeto.

Administra `TridentBackendConfig`backends usando `tridentctl`

```
`tridentctl` se puede usar para listar los backends que fueron creados
usando  `TridentBackendConfig`. Además, los administradores también pueden
elegir administrar completamente dichos backends a través de  `tridentctl`
eliminando  `TridentBackendConfig` y asegurándose de que
`spec.deletionPolicy` esté configurado como  `retain`.
```

Paso 0: identifica el backend

Por ejemplo, supongamos que el siguiente backend se creó usando `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 |
+------------------+----------------
+--------------------------------------+--------+---------+
```

Desde la salida, se puede ver que `TridentBackendConfig` fue creado exitosamente y está vinculado a un backend [observa el UUID del backend].

Paso 1: confirma que `deletionPolicy` esté configurado en `retain`

Echemos un vistazo al valor de `deletionPolicy`. Esto debe configurarse en `retain`. Esto asegura que cuando se elimina una CR de `TridentBackendConfig`, la definición del backend seguirá presente y podrás gestionarla con `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
```

No sigas con el siguiente paso a menos que `deletionPolicy` esté configurado en `retain`.

Paso 2: elimina el `TridentBackendConfig` CR

El último paso es eliminar el `TridentBackendConfig` CR. Después de confirmar que el `deletionPolicy` está configurado como `retain`, puedes continuar con la eliminación:

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

Al eliminar el `TridentBackendConfig` objeto, Trident simplemente lo elimina sin eliminar realmente el backend en sí.