# Usa Trident

Trident

NetApp
July 01, 2026

# Tabla de contenidos

# Usa Trident

## Prepara el nodo trabajador

Todos los nodos de trabajo del clúster de Kubernetes deben poder montar los volúmenes que has aprovisionado para tus pods. Para preparar los nodos de trabajo, tienes que instalar herramientas NFS, iSCSI, NVMe/TCP o FC según la selección de tu driver.

### Seleccionar las herramientas adecuadas

Si usas una combinación de controladores, deberías instalar todas las herramientas necesarias para tus controladores. Las versiones recientes de Red Hat Enterprise Linux CoreOS (RHCOS) ya traen las herramientas instaladas por defecto.

#### Herramientas NFS

"Instala las herramientas NFS" si usas: `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, o `azure-netapp-files`.

#### herramientas iSCSI

"Instala las herramientas iSCSI" si estás usando: `ontap-san`, `ontap-san-economy`, `solidfire-san`.

#### Herramientas NVMe

"Instala las herramientas NVMe" si estás usando `ontap-san` para nonvolatile memory express (NVMe) sobre TCP (NVMe/TCP).

> NetApp recomienda ONTAP 9.12 o versiones posteriores para NVMe/TCP.

#### Herramientas SCSI sobre FC

Consulta "Formas de configurar hosts SAN FC FC-NVMe" para más información sobre cómo configurar tus hosts SAN FC y FC-NVMe.

"Instala las herramientas FC" si estás usando `ontap-san` con sanType `fcp` (SCSI sobre FC).

**Puntos a tener en cuenta**: * SCSI sobre FC es compatible con los entornos OpenShift y KubeVirt. * SCSI sobre FC no es compatible con Docker. * La reparación automática de iSCSI no es aplicable a SCSI sobre FC.

#### Herramientas SMB

"Prepárate para aprovisionar volúmenes SMB" si estás usando: `ontap-nas` para aprovisionar volúmenes SMB.

### Descubrimiento de servicios de nodo

Trident intenta detectar automáticamente si el nodo puede ejecutar servicios iSCSI o NFS.

> El descubrimiento de servicios de nodo identifica los servicios descubiertos pero no garantiza que los servicios estén configurados correctamente. Por el contrario, la ausencia de un servicio descubierto no garantiza que el montaje del volumen falle.

#### Revisa los eventos

Trident crea eventos para el nodo para identificar los servicios descubiertos. Para revisar estos eventos, ejecuta:

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

**Revisa los servicios descubiertos**

Trident identifica los servicios habilitados para cada nodo en el Trident node CR. Para ver los servicios descubiertos, ejecuta:

```
tridentctl get node -o wide -n <Trident namespace>
```

## Volúmenes NFS

Instala las herramientas NFS usando los comandos para tu sistema operativo. Asegúrate de que el servicio de NFS se inicie durante el momento de arranque.

**RHEL 8+**

```
sudo yum install -y nfs-utils
```

**Ubuntu**

```
sudo apt-get install -y nfs-common
```

Reinicia tus nodos trabajadores después de instalar las herramientas NFS para evitar fallos al adjuntar volúmenes a los contenedores.

## volúmenes iSCSI

Trident puede establecer automáticamente una sesión iSCSI, escanear LUNs, descubrir dispositivos multivía, formatearlos y montarlos en un pod.

### capacidades de reparación automática iSCSI

Para los sistemas ONTAP, Trident ejecuta la reparación automática iSCSI cada cinco minutos para:

1. **Identifica** el estado de sesión iSCSI deseado y el estado de sesión iSCSI actual.
2. **Compara** el estado deseado con el estado actual para identificar las reparaciones necesarias. Trident determina las prioridades de reparación y cuándo anticipar las reparaciones.
3. **Realiza las reparaciones** necesarias para devolver el estado de sesión iSCSI actual al estado de sesión iSCSI deseado.

Los registros de la actividad de reparación automática se encuentran en el contenedor `trident-main` en el pod Daemonset correspondiente. Para ver los registros, tienes que haber establecido `debug` en "true" durante la instalación de Trident.

Las capacidades de reparación automática de Trident iSCSI pueden ayudar a prevenir:

- Sesiones iSCSI obsoletas o no saludables que podrían ocurrir después de un problema de conectividad de red. En el caso de una sesión obsoleta, Trident espera siete minutos antes de cerrar la sesión para restablecer la conexión con un portal.

  Por ejemplo, si se rotan los secretos CHAP en el controlador de almacenamiento y la red pierde conectividad, podrían persistir los secretos CHAP antiguos (*stale*). La reparación automática puede reconocer esto y restablecer automáticamente la sesión para aplicar los secretos CHAP actualizados.

- Faltan sesiones iSCSI
- LUNs perdidos

**Puntos a tener en cuenta antes de actualizar Trident**

- Si solo se utilizan igroups por nodo (introducidos en 23.04+), la reparación automática de iSCSI iniciará rescans SCSI para todos los dispositivos en el bus SCSI.
- Si solo se utilizan igroups con ámbito backend (obsoletos desde la 23.04), la reparación automática de iSCSI iniciará rescans SCSI para los IDs de LUN exactos en el bus SCSI.
- Si se utiliza una mezcla de igroups por nodo e igroups de ámbito backend, la reparación automática de iSCSI iniciará rescans SCSI para LUN IDs exactos en el bus SCSI.

### Instala las herramientas iSCSI

Instala las herramientas iSCSI usando los comandos para tu sistema operativo.

**Antes de empezar**

- Cada nodo en el clúster de Kubernetes debe tener un IQN único. **Este es un requisito previo necesario**.
- Si usas RHCOS versión 4.5 o posterior, u otra distribución de Linux compatible con RHEL, con el `solidfire-san` driver y Element OS 12.5 o anterior, asegúrate de que el algoritmo de autenticación CHAP esté configurado en MD5 en `/etc/iscsi/iscsid.conf`. Los algoritmos CHAP seguros compatibles con FIPS SHA1, SHA-256 y SHA3-256 están disponibles con Element 12.7.

```
sudo sed -i 's/^\(node.session.auth.chap_algs\).*/\1 = MD5/'
/etc/iscsi/iscsid.conf
```

- Cuando utilices nodos trabajadores que ejecutan RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) con iSCSI PVs, especifica el `discard` mountOption en la StorageClass para realizar la reclamación de espacio en línea. Consulta "Documentación de Red Hat".
- Asegúrate de que has actualizado a la última versión de `multipath-tools`.

**RHEL 8+**

1. Instala los siguientes paquetes del sistema:

```
sudo yum install -y lsscsi iscsi-initiator-utils device-mapper-
multipath
```

2. Verifica que la versión de iscsi-initiator-utils sea 6.2.0.874-2.el7 o posterior:

```
rpm -q iscsi-initiator-utils
```

3. Configura el escaneo en manual:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Activa la multivía:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Asegúrate de que `/etc/multipath.conf` contiene `find_multipaths no` bajo `defaults`.

5. Asegúrate de que `iscsid` y `multipathd` están en funcionamiento:

```
sudo systemctl enable --now iscsid multipathd
```

6. Activa e inicia `iscsi`:

```
sudo systemctl enable --now iscsi
```

**Ubuntu**

1. Instala los siguientes paquetes del sistema:

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. Verifica que la versión de open-iscsi sea 2.0.874-5ubuntu2.10 o posterior (para bionic) o 2.0.874-7.1ubuntu6.1 o posterior (para focal):

```
dpkg -l open-iscsi
```

3. Configura el escaneo en manual:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Activa la multivía:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> Asegúrate de que `/etc/multipath.conf` contiene `find_multipaths no` bajo `defaults`.

5. Asegúrate de que `open-iscsi` y `multipath-tools` están habilitados y en funcionamiento:

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

> Para Ubuntu 18.04, debes descubrir los puertos de destino con `iscsiadm` antes de iniciar `open-iscsi` para que el demonio iSCSI se inicie. También puedes modificar el servicio `iscsi` para que se inicie `iscsid` automáticamente.

**Configura o desactiva la reparación automática de iSCSI**

Puedes configurar las siguientes opciones de reparación automática de Trident iSCSI para arreglar las sesiones obsoletas:

- **intervalo de reparación automática iSCSI**: Determina la frecuencia con la que se invoca la reparación automática iSCSI (por defecto: 5 minutos). Puedes configurarlo para que se ejecute con más frecuencia estableciendo un número menor o con menos frecuencia estableciendo un número mayor.

Establecer el intervalo de reparación automática iSCSI en 0 detiene la reparación automática iSCSI por completo. No recomendamos desactivar la reparación automática iSCSI; solo debería desactivarse en ciertos escenarios cuando la reparación automática iSCSI no funciona como se espera o para fines de depuración.

- **tiempo de espera de reparación automática iSCSI**: determina la duración de la espera de la reparación automática iSCSI antes de cerrar la sesión no saludable e intentar volver a iniciar sesión (por defecto: 7 minutos). Puedes configurarlo con un número mayor para que las sesiones que se identifiquen como no saludables tengan que esperar más tiempo antes de cerrar la sesión y luego se intente volver a iniciar sesión, o con un número menor para cerrar la sesión e iniciarla antes.

**Helm**

Para configurar o cambiar los ajustes de reparación automática de iSCSI, pasa los parámetros `iscsiSelfHealingInterval` y `iscsiSelfHealingWaitTime` durante la instalación o actualización de helm.

El siguiente ejemplo establece el intervalo de reparación automática iSCSI en 3 minutos y el tiempo de espera de reparación automática en 6 minutos:

```
helm install trident trident-operator-100.2602.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

**tridentctl**

Para configurar o cambiar los ajustes de reparación automática de iSCSI, pasa los parámetros `iscsi-self-healing-interval` y `iscsi-self-healing-wait-time` durante la instalación o actualización de tridentctl.

El siguiente ejemplo establece el intervalo de reparación automática iSCSI en 3 minutos y el tiempo de espera de reparación automática en 6 minutos:

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

## Volúmenes NVMe/TCP

Instala las herramientas NVMe usando los comandos para tu sistema operativo.

- NVMe requiere RHEL 9 o posterior.
- Si la versión del kernel de tu nodo de Kubernetes es demasiado antigua o si el paquete NVMe no está disponible para tu versión del kernel, puede que tengas que actualizar la versión del kernel de tu nodo a una que tenga el paquete NVMe.

**RHEL 9**

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

**Ubuntu**

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

### Verifica la instalación

Después de la instalación, verifica que cada nodo del clúster de Kubernetes tenga un NQN único usando el comando:

```
cat /etc/nvme/hostnqn
```

> Trident modifica el valor de `ctrl_device_tmo` para asegurarse de que NVMe no abandone la ruta si se cae. No cambies esta configuración.

## Volúmenes SCSI sobre FC

Ahora puedes usar el protocolo Fibre Channel (FC) con Trident para aprovisionar y gestionar recursos de almacenamiento en el sistema ONTAP.

### Prerrequisitos

Configura los ajustes de red y de nodo necesarios para FC.

#### Configuración de red

1. Obtén el WWPN de las interfaces de destino. Consulta "network interface show" para más información.
2. Obtén el WWPN de las interfaces en el iniciador (host).

   Consulta las utilidades correspondientes del sistema operativo host.

3. Configura la división en zonas en el conmutador FC usando los WWPN del host y del target.

   Consulta la documentación del proveedor del conmutador correspondiente para más información.

   Consulta la siguiente documentación de ONTAP para más detalles:

   - "Visión general de la división en zonas Fibre Channel y FCoE"
   - "Formas de configurar hosts SAN FC FC-NVMe"

### Instala las herramientas FC

Instala las herramientas FC usando los comandos para tu sistema operativo.

- Cuando uses nodos trabajadores que ejecutan RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) con FC PVs, especifica el `discard` mountOption en la StorageClass para realizar la reclamación de espacio en línea. Consulta "Documentación de Red Hat".

**RHEL 8+**

1. Instala los siguientes paquetes del sistema:

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. Activa la multivía:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Asegúrate de que `/etc/multipath.conf` contiene `find_multipaths no` bajo `defaults`.

3. Asegúrate de que `multipathd` está funcionando:

```
sudo systemctl enable --now multipathd
```

**Ubuntu**

1. Instala los siguientes paquetes del sistema:

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. Activa la multivía:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> Asegúrate de que `/etc/multipath.conf` contiene `find_multipaths no` bajo `defaults`.

3. Asegúrate de que `multipath-tools` está activado y en funcionamiento:

```
sudo systemctl status multipath-tools
```

## Prepárate para aprovisionar volúmenes SMB

Puedes aprovisionar volúmenes SMB usando `ontap-nas` drivers.

> Debes configurar tanto los protocolos NFS como SMB/CIFS en la SVM para crear un `ontap-nas-economy` volumen SMB para clústeres ONTAP locales. Si no configuras alguno de estos protocolos, la creación del volumen SMB fallará.

> `autoExportPolicy` no es compatible con volúmenes SMB.

**Antes de empezar**

Antes de que puedas aprovisionar volúmenes SMB, debes tener lo siguiente.

- Un clúster de Kubernetes con un nodo controlador Linux y al menos un nodo trabajador Windows que ejecuta Windows Server 2022. Trident solo admite volúmenes SMB montados en pods que se ejecutan en nodos Windows.
- Al menos un secreto de Trident con tus credenciales de Active Directory. Para generar un secreto `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Un proxy CSI configurado como un servicio de Windows. Para configurar un `csi-proxy`, consulta "GitHub: CSI Proxy" o "GitHub: CSI Proxy para Windows" para nodos de Kubernetes que se ejecutan en Windows.

**Pasos**

1. Para ONTAP local, puedes crear opcionalmente un recurso compartido SMB o Trident puede crear uno para ti.

   > Se requieren recursos compartidos SMB para Amazon FSx for ONTAP.

   Puedes crear los recursos compartidos de administrador de SMB de dos maneras: usando el complemento "Microsoft Management Console" Shared Folders o usando la CLI de ONTAP. Para crear los recursos compartidos de SMB usando la CLI de ONTAP:

   a. Si es necesario, crea la estructura de ruta del directorio para el recurso compartido.

      El `vserver cifs share create`comando comprueba la ruta especificada en la opción -path durante la creación del recurso compartido. Si la ruta especificada no existe, el comando falla.

   b. Crea un recurso compartido SMB asociado con la SVM especificada:

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

c. Verifica que se creó el recurso compartido:

```
vserver cifs share show -share-name share_name
```

Consulta "Crear un recurso compartido SMB" para obtener detalles completos.

2. Al crear el backend, debes configurar lo siguiente para especificar los volúmenes SMB. Para todas las opciones de configuración del backend de FSx for ONTAP, consulta "Opciones de configuración de FSx for ONTAP y ejemplos".

| Parámetro | Descripción | Ejemplo |
|---|---|---|
| `smbShare` | Puedes especificar uno de los siguientes: el nombre de un recurso compartido SMB creado usando Microsoft Management Console o la CLI de ONTAP, un nombre para permitir que Trident cree el recurso compartido SMB, o puedes dejar el parámetro en blanco para evitar el acceso compartido común a los volúmenes. Este parámetro es opcional para ONTAP local. Este parámetro es obligatorio para los backends de Amazon FSx for ONTAP y no puede estar en blanco. | `smb-share` |
| `nasType` | **Debe establecerse en `smb`.** Si es nulo, el valor predeterminado es `nfs`. | `smb` |
| `securityStyle` | Estilo de seguridad para nuevos volúmenes. **Debe configurarse en `ntfs` o `mixed` para volúmenes SMB.** | `ntfs` o `mixed` para volúmenes SMB |
| `unixPermissions` | Modo para nuevos volúmenes. **Debe dejarse vacío para volúmenes SMB.** | "" |

# Configura y gestiona backends

## Configura backends

Un backend define la relación entre Trident y un sistema de almacenamiento. Le dice a Trident cómo comunicarse con ese sistema de almacenamiento y cómo Trident debe aprovisionar volúmenes desde él.

Trident ofrece automáticamente grupos de almacenamiento de backends que cumplen los requisitos definidos por una clase de almacenamiento. Aprende cómo configurar el backend para tu sistema de almacenamiento.

- "Configura un backend de Azure NetApp Files"
- "Configura un backend de Google Cloud NetApp Volumes"
- "Configura un backend de NetApp HCI o SolidFire"
- "Configura un backend con controladores NAS ONTAP o Cloud Volumes ONTAP"
- "Configura un backend con controladores SAN ONTAP o Cloud Volumes ONTAP"

- "Usa Trident con Amazon FSx for NetApp ONTAP"

## Azure NetApp Files

### Configura un backend de Azure NetApp Files

Usa Azure NetApp Files como backend para Trident. Este backend admite volúmenes NFS y SMB. Trident admite identidades gestionadas e identidad de carga de trabajo para clústeres de Azure Kubernetes Service (AKS).

#### Entornos cloud de Azure compatibles

Trident es compatible con los backends de Azure NetApp Files en varios entornos cloud de Azure.

Las nubes de Azure compatibles incluyen:

- Azure Commercial
- Azure Government (Azure Government / MAG)

Cuando despliegues Trident o configures un backend de Azure NetApp Files, asegúrate de que Azure Resource Manager y los endpoints de autenticación coincidan con tu entorno de cloud de Azure.

#### Revisa el soporte del controlador de Azure NetApp Files

Trident proporciona el siguiente controlador de almacenamiento Azure NetApp Files.

Los modos de acceso admitidos incluyen *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX) y *ReadWriteOncePod* (RWOP).

| Controlador | Protocolo | volumeMode | Modos de acceso admitidos | Sistemas de archivos compatibles |
|---|---|---|---|---|
| `azure-netapp-files` | NFS SMB | Sistema de archivos | RWO, ROX, RWX, RWOP | `nfs`, `smb` |

#### Revisa las consideraciones

- Azure NetApp Files no admite volúmenes menores de 50 GiB. Trident crea un volumen de 50 GiB cuando se solicita un volumen menor.
- Trident solo admite volúmenes SMB montados en pods que se ejecutan en nodos Windows.
- Las implementaciones de Azure NetApp Files en nubes Azure no comerciales requieren puntos finales de Azure Resource Manager y de autenticación específicos para cada nube. Asegúrate de que Trident y cualquier configuración de backend usen los puntos finales apropiados para tu entorno de nube Azure.

#### Usa identidades gestionadas para AKS

Trident es compatible con "identidades gestionadas" para clústeres AKS.

Si usas `tridentctl` para crear o gestionar backends de Azure NetApp Files, asegúrate de que esté configurado para el entorno de cloud de Azure correcto.

Para usar identidades gestionadas, debes tener:

- Un clúster de Kubernetes implementado mediante AKS
- Identidades gestionadas configuradas en el clúster de Kubernetes de AKS
- Trident instalado con `cloudProvider` configurado en `"Azure"`

**Operador de Trident**

Edita `tridentorchestrator_cr.yaml` y establece `cloudProvider` en `"Azure"`.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

**Helm**

El siguiente ejemplo instala Trident y configura `cloudProvider` usando la variable de entorno `$CP`:

```
helm install trident trident-operator-100.2602.0.tgz --create-namespace
--namespace <trident-namespace> --set cloudProvider=$CP
```

**<code>tridentctl</code>**

El siguiente ejemplo instala Trident y establece la `cloud-provider` bandera en `Azure`:

```
tridentctl install --cloud-provider="Azure" -n trident
```

#### Usa la identidad de carga de trabajo para AKS

La identidad de carga de trabajo permite que los pods de Kubernetes accedan a los recursos de Azure autenticándose como una identidad de carga de trabajo.

Si usas `tridentctl` para crear o gestionar backends de Azure NetApp Files, asegúrate de que esté configurado para el entorno de cloud de Azure correcto.

Para usar la identidad de carga de trabajo, debes tener:

- Un clúster de Kubernetes implementado mediante AKS
- Identidad de carga de trabajo y oidc-issuer configurados en el clúster AKS Kubernetes
- Trident instalado con `cloudProvider` configurado en `"Azure"` y `cloudIdentity` configurado en el valor de identidad de la carga de trabajo

**Operador de Trident**

Edita `tridentorchestrator_cr.yaml` y establece `cloudProvider` en `"Azure"`. Establece `cloudIdentity` en `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx' # Edit
```

**Helm**

Establece los valores para las opciones **cloud-provider (CP)** y **cloud-identity (CI)** usando las siguientes variables de entorno:

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx'"
```

El siguiente ejemplo instala Trident y establece `cloudProvider` usando `$CP` y establece `cloudIdentity` usando `$CI`:

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

**<code>tridentctl</code>**

Establece los valores para las banderas **cloud provider** y **cloud identity** usando las siguientes variables de entorno:

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx"
```

El siguiente ejemplo instala Trident y establece `cloud-provider` en `$CP` y `cloud-identity` en `$CI`:

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

### Prepárate para configurar un backend de Azure NetApp Files

Antes de que puedas configurar tu backend de Azure NetApp Files, necesitas asegurarte de que se cumplan los siguientes requisitos.

#### Entornos cloud de Azure compatibles

Trident es compatible con los backends de Azure NetApp Files en varios entornos cloud de Azure.

Las nubes de Azure compatibles incluyen:

- Azure Commercial
- Azure Government (Azure Government / MAG)

Cuando prepares tu entorno, asegúrate de que tu suscripción de Azure, la configuración de identidad y los recursos de Azure NetApp Files se creen en el entorno de cloud de Azure adecuado.

#### Requisitos previos para volúmenes NFS y SMB

Si estás usando Azure NetApp Files por primera vez o en una nueva ubicación, necesitas hacer una configuración inicial para preparar Azure NetApp Files y crear un volumen NFS. Consulta "Azure: configura Azure NetApp Files y crea un volumen NFS".

Para configurar y usar un "Azure NetApp Files"backend, necesitas lo siguiente:

> - `subscriptionID`, `tenantID`, `clientID`, `location` y `clientSecret` son opcionales cuando usas identidades administradas en un clúster de AKS.
> - `tenantID`, `clientID` y `clientSecret` son opcionales cuando usas una identidad en la nube en un clúster de AKS.
> - Las implementaciones de Azure NetApp Files en nubes Azure no comerciales requieren puntos finales de Azure Resource Manager y de autenticación específicos para cada nube. Asegúrate de que Trident y cualquier configuración de backend usen los puntos finales apropiados para tu entorno de nube Azure.

- Un fondo de capacidad. Consulta "Microsoft: crea un grupo de capacidad para Azure NetApp Files".
- Una subred delegada a Azure NetApp Files. Consulta "Microsoft: delega una subred a Azure NetApp Files".
- `subscriptionID` desde una suscripción de Azure con Azure NetApp Files habilitado.
- `tenantID`, `clientID` y `clientSecret` de un "Registro de la aplicación" en Azure Active Directory con permisos suficientes para el servicio Azure NetApp Files. El registro de la aplicación debe usar una de las siguientes opciones:
  - El rol de Owner o Contributor "predefinido por Azure".
  - A "rol de colaborador personalizado" en el nivel de suscripción (`assignableScopes`) con los siguientes permisos que están limitados solo a lo que Trident requiere. Después de crear el rol personalizado, "asigna el rol usando el portal de Azure".

**Rol de colaborador personalizado**

```
{
  "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.NetApp/netAppAccounts/capacityPools/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/delete",
```

```
            "Microsoft.Features/features/read",
            "Microsoft.Features/operations/read",
            "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
          ],
          "notActions": [],
          "dataActions": [],
          "notDataActions": []
        }
      ]
    }
}
```

- El Azure `location` que contiene al menos un "subred delegada". Desde Trident 22.01, el parámetro `location` es un campo obligatorio en el nivel superior del archivo de configuración del backend. Se ignoran los valores de ubicación especificados en los grupos virtuales.
- Para usar `Cloud Identity`, consigue el `client ID` de un "identidad administrada asignada por el usuario" y especifica ese ID en `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

#### Requisitos adicionales para volúmenes SMB

Para crear un volumen SMB, debes tener:

- Active Directory configurado y conectado a Azure NetApp Files. Consulta "Microsoft: crear y administrar conexiones de Active Directory para Azure NetApp Files".
- Un clúster de Kubernetes con un nodo controlador Linux y al menos un nodo trabajador Windows que ejecuta Windows Server 2022. Trident solo admite volúmenes SMB montados en pods que se ejecutan en nodos Windows.
- Al menos un secreto de Trident que contenga tus credenciales de Active Directory para que Azure NetApp Files pueda autenticarse en Active Directory. Para generar un secreto `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Un proxy CSI configurado como un servicio de Windows. Para configurar un `csi-proxy`, consulta "GitHub: CSI Proxy" o "GitHub: CSI Proxy para Windows" para nodos de Kubernetes que se ejecutan en Windows.

### Opciones y ejemplos de configuración del backend de Azure NetApp Files

Conoce las opciones de configuración de backend de NFS y SMB para Azure NetApp Files y revisa ejemplos de configuración.

#### Opciones de configuración del backend

Trident utiliza tu configuración de backend (subred, red virtual, nivel de servicio y ubicación) para crear volúmenes de Azure NetApp Files en pools de capacidad que están disponibles en la ubicación solicitada y coinciden con el nivel de servicio y la subred solicitados.

Los backends de Azure NetApp Files proporcionan estas opciones de configuración.

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `version` | Versión de configuración del backend. | Siempre 1 |
| `storageDriverName` | Nombre del controlador de almacenamiento | "azure-netapp-files" |
| `backendName` | Nombre personalizado para el backend de almacenamiento | Nombre del driver + "_" + caracteres aleatorios |
| `subscriptionID` | El identificador de suscripción de tu suscripción de Azure opcional cuando las identidades administradas están habilitadas en un clúster de AKS. | |
| `tenantID` | El ID del inquilino de un App Registration es opcional cuando se usan identidades administradas o identidad en la nube en un clúster de AKS. | |
| `clientID` | El ID de cliente de un registro de aplicación, opcional cuando se usan identidades administradas o identidad en la nube en un clúster de AKS. | |
| `clientSecret` | El secreto del cliente de un registro de aplicación es opcional cuando se usan identidades administradas o identidad en la nube en un clúster de AKS. | |
| `serviceLevel` | Uno de `Standard`, `Premium` o `Ultra` | "" (aleatorio) |
| `location` | Nombre de la ubicación de Azure donde se crearán los nuevos volúmenes Opcional cuando las identidades administradas están habilitadas en un clúster de AKS. | |

| Parámetro | Descripción | Predeterminado |
| --- | --- | --- |
| `resourceGroups` | Lista de grupos de recursos para filtrar recursos descubiertos | "[]" (sin filtro) |
| `netappAccounts` | Lista de cuentas de NetApp para filtrar recursos descubiertos | "[]" (sin filtro) |
| `capacityPools` | Lista de grupos de capacidad para filtrar recursos descubiertos | "[]" (sin filtro, aleatorio) |
| `virtualNetwork` | Nombre de una red virtual con una subred delegada | "" |
| `subnet` | Nombre de una subred delegada a `Microsoft.Netapp/volumes` | "" |
| `networkFeatures` | Conjunto de funciones de VNet para un volumen, puede ser `Basic` o `Standard`. Network Features no está disponible en todas las regiones y podría tener que habilitarse en una suscripción. Especificar `networkFeatures` cuando la funcionalidad no está habilitada hace que falle el aprovisionamiento del volumen. | "" |
| `nfsMountOptions` | Control detallado de las opciones de montaje de NFS. Ignorado para volúmenes SMB. Para montar volúmenes usando la versión de NFS 4.1, incluye `nfsvers=4` en la lista de opciones de montaje separadas por comas para elegir NFS v4.1. Las opciones de montaje establecidas en una definición de clase de almacenamiento reemplazan las opciones de montaje establecidas en la configuración de backend. | "nfsvers=3" |
| `limitVolumeSize` | Falla el aprovisionamiento si el tamaño del volumen solicitado es superior a este valor | "" (no aplicado por defecto) |
| `debugTraceFlags` | Indicadores de depuración para utilizar cuando estés solucionando problemas. Ejemplo, `\{"api": false, "method": true, "discovery": true}`. No uses esto a menos que estés resolviendo problemas y necesites un volcado detallado del registro. | null |

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `nasType` | Configura la creación de volúmenes NFS o SMB. Las opciones son `nfs`, `smb` o null. Si lo configuras en null, se usarán volúmenes NFS por defecto. | `nfs` |
| `supportedTopologies` | Representa una lista de regiones y zonas compatibles con este backend. Para obtener más información, consulta "Usa la topología CSI". | |
| `qosType` | Representa el tipo de QoS: automático o manual. | Automático |
| `maxThroughput` | Establece el rendimiento máximo permitido en MiB/s. Compatible solo con grupos de capacidad de QoS manual. | `4 MiB/sec` |

Para obtener más información sobre las funciones de red, consulta "Configura funciones de red para un volumen de Azure NetApp Files".

#### Considera los entornos cloud de Azure (26.02)

A partir de la versión 26.02, Trident admite la creación y gestión de backends de Azure NetApp Files en varios entornos cloud de Azure.

Las nubes de Azure compatibles incluyen:

- Azure Commercial
- Azure Government (Azure Government / MAG)

Cuando despliegas Trident o creas un backend de Azure NetApp Files, asegúrate de que Azure Resource Manager y los endpoints de autenticación coincidan con tu entorno de Azure cloud. Si los endpoints no coinciden, `tridentctl` no puede autenticarse y la creación del backend falla.

### Permisos y recursos necesarios

Si recibes el error "No capacity pools found" al crear un PVC, es probable que el registro de tu aplicación no tenga los permisos y recursos necesarios (subred, red virtual, capacity pool) asociados. Si la depuración está activada, Trident registra los recursos de Azure descubiertos cuando se crea el backend. Verifica que se esté usando un rol adecuado.

Los valores para `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork` y `subnet` se pueden especificar usando nombres cortos o nombres completamente calificados. Se recomiendan los nombres completamente calificados en la mayoría de las situaciones, ya que los nombres cortos pueden coincidir con varios recursos con el mismo nombre.

Si la vNet está ubicada en un grupo de recursos diferente de la cuenta de almacenamiento de Azure NetApp Files (ANF), especifica el grupo de recursos para la red virtual mientras configuras la lista resourceGroups para el backend.

Los `resourceGroups`, `netappAccounts`, y `capacityPools` valores son filtros que restringen el conjunto de recursos detectados a los disponibles para este backend de almacenamiento y pueden especificarse en cualquier combinación. Los nombres completos siguen este formato:

| Tipo | Formato |
|---|---|
| Grupo de recursos | <resource group> |
| Cuenta de NetApp | <resource group>/<netapp account> |
| Fondo de capacidad | <resource group>/<netapp account>/<capacity pool> |
| Red virtual | <resource group>/<virtual network> |
| Subred | <resource group>/<virtual network>/<subnet> |

### Aprovisionamiento de volumen

Puedes controlar el aprovisionamiento de volúmenes predeterminados especificando las siguientes opciones en una sección especial del archivo de configuración. Consulta Configuraciones de ejemplo para más detalles.

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `exportRule` | Reglas de exportación para volúmenes nuevos. `exportRule` debe ser una lista separada por comas de cualquier combinación de direcciones IPv4 o subredes IPv4 en notación CIDR. Ignorado para volúmenes SMB. | "0.0.0.0/0" |
| `snapshotDir` | Acceso al `.snapshot` directorio | `true`, `false` (establecer explícitamente). |
| `size` | El tamaño predeterminado de los nuevos volúmenes | "100G" |
| `unixPermissions` | Los permisos unix de nuevos volúmenes (4 dígitos octales). Ignorado para volúmenes SMB. | "" (función de vista previa, requiere whitelisting en la suscripción) |

#### Configuraciones de ejemplo

Los siguientes ejemplos muestran configuraciones básicas que dejan la mayoría de los parámetros en sus valores predeterminados. Esta es la forma más fácil de definir un backend.

#### Configuración mínima

Esta es la configuración mínima absoluta del backend. Con esta configuración, Trident detecta todas tus cuentas NetApp, grupos de capacidad y subredes delegadas a Azure NetApp Files en la ubicación configurada, y coloca los nuevos volúmenes en uno de esos grupos y subredes aleatoriamente. Porque `nasType` se omite, la `nfs` configuración predeterminada se aplica y el backend aprovisionará volúmenes NFS.

Esta configuración es ideal cuando recién estás empezando a usar Azure NetApp Files y estás probando cosas, pero en la práctica vas a querer proporcionar un alcance adicional para los volúmenes que aprovisiones.

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
  tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
  clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
  clientSecret: SECRET
  location: eastus
```

**Identidades administradas para AKS**

Esta configuración de backend omite `subscriptionID`, `tenantID`, `clientID` y `clientSecret`, que son opcionales cuando usas identidades administradas.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - resource-group-1/netapp-account-1/ultra-pool
  resourceGroups:
    - resource-group-1
  netappAccounts:
    - resource-group-1/netapp-account-1
  virtualNetwork: resource-group-1/eastus-prod-vnet
  subnet: resource-group-1/eastus-prod-vnet/eastus-anf-subnet
```

#### Identidad en la nube para AKS

Esta configuración de backend omite `tenantID`, `clientID` y `clientSecret`, que son opcionales cuando usas una identidad en la nube.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

#### Configuración específica del nivel de servicio con filtros de grupo de capacidad

Esta configuración de backend coloca volúmenes en la `eastus` ubicación de Azure en un `Ultra` grupo de capacidad. Trident detecta automáticamente todas las subredes delegadas a Azure NetApp Files en esa ubicación y coloca un nuevo volumen en una de ellas aleatoriamente.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
```

#### Ejemplo de backend con grupos de capacidad QoS manuales

Esta configuración de back-end coloca volúmenes en la `eastus` ubicación de Azure con grupos de capacidad de QoS manuales.

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

**Configuración avanzada**

Esta configuración de backend reduce aún más el alcance de la ubicación del volumen a una sola subred, y también modifica algunos valores predeterminados de aprovisionamiento de volumen.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
virtualNetwork: application-group-1/eastus-prod-vnet
subnet: application-group-1/eastus-prod-vnet/my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: "true"
  size: 200Gi
  unixPermissions: "0777"
```

#### Configuración de virtual pool

Esta configuración de backend define varios grupos de almacenamiento en un solo archivo. Esto es útil cuando tienes varios grupos de capacidad que admiten diferentes niveles de servicio y quieres crear clases de almacenamiento en Kubernetes que los representen. Se usaron etiquetas de grupo virtual para diferenciar los grupos según `performance`.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
  - application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
  - labels:
      performance: gold
    serviceLevel: Ultra
    capacityPools:
      - application-group-1/netapp-account-1/ultra-1
      - application-group-1/netapp-account-1/ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - application-group-1/netapp-account-1/premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - application-group-1/netapp-account-1/standard-1
      - application-group-1/netapp-account-1/standard-2
```

#### Configuración de topologías admitidas

Trident facilita el aprovisionamiento de volúmenes para cargas de trabajo según regiones y zonas de disponibilidad. El `supportedTopologies` bloque en esta configuración de backend se usa para proporcionar una lista de regiones y zonas por backend. Los valores de región y zona especificados aquí deben coincidir con los valores de región y zona de las etiquetas en cada nodo del clúster de Kubernetes. Estas regiones y zonas representan la lista de valores permitidos que se pueden proporcionar en una clase de almacenamiento. Para las clases de almacenamiento que contienen un subconjunto de las regiones y zonas proporcionadas en un backend, Trident crea volúmenes en la región y zona mencionadas. Para obtener más información, consulta "Usa la topología CSI".

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
supportedTopologies:
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-1
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-2
```

#### Definiciones de clases de almacenamiento

Las siguientes `StorageClass` definiciones se refieren a los grupos de almacenamiento anteriores.

### Ejemplos de definiciones usando `parameter.selector` field

Usando `parameter.selector` puedes especificar para cada `StorageClass`el pool virtual que se usa para alojar un volumen. El volumen tendrá los aspectos definidos en el pool elegido.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold
allowVolumeExpansion: true

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
allowVolumeExpansion: true

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze
allowVolumeExpansion: true
```

#### Definiciones de ejemplo para volúmenes SMB

Usando `nasType`, `node-stage-secret-name` y `node-stage-secret-namespace`, puedes especificar un volumen SMB y proporcionar las credenciales de Active Directory necesarias.

**Configuración básica en el espacio de nombres predeterminado**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

**Usando diferentes secretos por espacio de nombres**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

**Usando diferentes secretos por volumen**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb`filtros para grupos que admiten volúmenes SMB. `nasType: nfs` o `nasType: null`filtros para grupos NFS.

#### Crea el backend

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

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

Si usas una nube Azure no comercial, asegúrate de que `tridentctl` esté configurado para usar los endpoints de Azure Resource Manager y autenticación para tu entorno de nube Azure. Si falla la creación del backend, revisa tu configuración del backend y mira los registros para ver cuál es la causa:

```
tridentctl logs
```

Después de identificar y corregir el problema con el archivo de configuración, puedes volver a ejecutar el comando create.

## Google Cloud NetApp Volumes

### Configura Google Cloud NetApp Volumes

Puedes configurar Google Cloud NetApp Volumes como backend para que Trident aprovisione almacenamiento para cargas de trabajo de Kubernetes.

#### Descripción general

Trident es compatible con Google Cloud NetApp Volumes tanto para cargas de trabajo NAS (NFS y SMB) como en bloque (iSCSI).

- Las cargas de trabajo NAS utilizan el `google-cloud-netapp-volumes` backend
- Las cargas de trabajo en bloque (iSCSI) utilizan el `google-cloud-netapp-volumes-san` backend

Los volúmenes NAS proporcionan almacenamiento basado en archivos y se accede a ellos mediante protocolos NFS o SMB. Estos volúmenes admiten el acceso compartido a través de varios pods o nodos.

Los volúmenes en bloque proporcionan almacenamiento en bloque sin procesar y se accede a ellos como dispositivos iSCSI conectados a nodos Kubernetes. Estos volúmenes se utilizan cuando las aplicaciones requieren acceso a nivel de bloque.

Esto se aplica a los siguientes entornos:

- Trident 26.02 y posteriores
- Google Kubernetes Engine (GKE) o Red Hat OpenShift
- Pools de almacenamiento de Google Cloud NetApp Volumes

Para configurar el almacenamiento en bloque (iSCSI), consulta "Configura el almacenamiento en bloque (iSCSI)".

#### Prepárate para configurar

La identidad en la nube permite que las cargas de trabajo de Kubernetes accedan a los recursos de Google Cloud autenticándose como una identidad de carga de trabajo en vez de usar credenciales estáticas.

Para usar la identidad en la nube con Google Cloud NetApp Volumes, debes tener:

- Un clúster de Kubernetes desplegado usando Google Kubernetes Engine (GKE)
- Identidad de carga de trabajo habilitada en el clúster GKE y el servidor de metadatos habilitado en los grupos de nodos
- Una cuenta de servicio de Google Cloud con la función de Google Cloud NetApp Volumes Admin (`roles/netapp.admin`) o una función personalizada equivalente
- Trident instalado con el cloud provider establecido en `GCP` y la cloud identity annotation configurada

**Operador de Trident**

Para instalar Trident utilizando el operador Trident, edita `tridentorchestrator_cr.yaml`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  cloudProvider: "GCP"
  cloudIdentity: "iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com"
```

**Helm**

Establece el proveedor de la nube y la identidad de la nube al instalar Trident con Helm:

```
helm install trident trident-operator-100.6.0.tgz \
  --set cloudProvider=GCP \
  --set cloudIdentity="iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com"
```

**tridentctl**

Instala Trident especificando el proveedor de la nube y la identidad de la nube:

```
tridentctl install \
  --cloud-provider=GCP \
  --cloud-identity="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com" \
  -n trident
```

#### Configurar el almacenamiento NAS

> Para los grupos de almacenamiento UNIFIED de Google Cloud NetApp Volumes, Trident aplica reglas de nomenclatura y validación específicas de UNIFIED durante las operaciones de volumen.

Al localizar un volumen, Trident puede evaluar múltiples variantes de nombre de volumen compatibles (por ejemplo, formatos de guion y guion bajo) para mejorar la fiabilidad de la importación y el descubrimiento.

### Detalles del controlador

Trident proporciona el `google-cloud-netapp-volumes` controlador para aprovisionar almacenamiento NAS desde Google Cloud NetApp Volumes.

El controlador admite los siguientes modos de acceso:

- ReadWriteOnce (RWO)
- ReadOnlyMany (ROX)
- ReadWriteMany (RWX)
- ReadWriteOncePod (RWOP)

| Controlador | Protocolo | volumeMode | Modos de acceso admitidos | Sistemas de archivos compatibles |
|---|---|---|---|---|
| `google-cloud-netapp-volumes` | NFS SMB | Sistema de archivos | RWO, ROX, RWX, RWOP | `nfs`, `smb` |

### Configura un backend NAS Trident

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
```

### Aprovisionar volúmenes NAS

Los volúmenes NAS se aprovisionan usando el `google-cloud-netapp-volumes` backend y son compatibles con los protocolos NFS y SMB.

### StorageClass para volúmenes NFS

Para aprovisionar volúmenes NFS, establece `nasType` en `nfs`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "nfs"
allowVolumeExpansion: true
```

#### StorageClass para volúmenes SMB

Para aprovisionar volúmenes SMB, establece `nasType` en `smb` y proporciona las credenciales.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
allowVolumeExpansion: true
```

#### Ejemplo de PersistentVolumeClaim (RWX)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwx
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

#### Ejemplo de PersistentVolumeClaim (RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

Los volúmenes NAS usan `volumeMode: Filesystem`.

### Configura Google Cloud NetApp Volumes para cargas de trabajo SAN

Puedes configurar Trident para aprovisionar volúmenes de almacenamiento en bloque usando el protocolo iSCSI desde Google Cloud NetApp Volumes. Los volúmenes SAN se aprovisionan desde grupos de almacenamiento Flex Unified usando el controlador de almacenamiento `google-cloud-netapp-volumes-san`.

Este controlador está dedicado a cargas de trabajo en bloque y no es compatible con protocolos NAS.

El `google-cloud-netapp-volumes-san` backend es necesario para aprovisionar volúmenes de bloques iSCSI. El `google-cloud-netapp-volumes` backend admite solo protocolos NAS y no puede usarse para cargas de trabajo SAN.

#### Descripción general

Trident es compatible con las cargas de trabajo SAN (iSCSI) de Google Cloud NetApp Volumes mediante el controlador `google-cloud-netapp-volumes-san`.

Los volúmenes SAN se aprovisionan desde grupos de almacenamiento Flex Unified y se presentan a los nodos de Kubernetes como dispositivos de bloques iSCSI.

Esto se aplica a los siguientes entornos:

- Trident 26.02 y posteriores
- Google Kubernetes Engine (GKE) o Red Hat OpenShift
- Pools de almacenamiento unificado de Google Cloud NetApp Volumes Flex
- cargas de trabajo basadas en iSCSI

#### Pools de almacenamiento unificado Flex

Los grupos de almacenamiento de Flex Unified proporcionan almacenamiento en bloque mediante el protocolo iSCSI y son necesarios para el aprovisionamiento de SAN:

- Se admiten los pools REGIONALES de Flex Unified.
- Las agrupaciones de Flex Unified ZONAL son compatibles a partir de Trident 26.02.1.
- Solo se admite el nivel de servicio **Flex** para cargas de trabajo SAN.

#### Configura un backend SAN Trident

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-san
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
      performance: flex
    network: "<vpc-network>"
    serviceLevel: Flex
```

#### Crea un StorageClass

Después de configurar el backend SAN, crea un StorageClass que haga referencia al `google-cloud-netapp-volumes-san` driver.

El tipo de sistema de archivos se define en el StorageClass, no en el backend.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Tipos de sistemas de archivos compatibles:

- `ext4` (predeterminado)
- `ext3`

- `xfs`

> El controlador SAN solo es compatible con el nivel de servicio Flex y no utiliza parámetros de backend específicos de NAS como `exportRule`, `unixPermissions`, `nasType`, `snapshotDir`, `nfsMountOptions` o configuraciones relacionadas con el tiering.

#### Provisiona volúmenes en bloque

##### ReadWriteOnce (RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

##### ReadWriteOncePod (RWOP)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwop
spec:
  accessModes:
    - ReadWriteOncePod
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

##### ReadOnlyMany (ROX)

Un patrón común para ROX es clonar un volumen ReadWriteOnce existente y montar el clon como de solo lectura.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rox
spec:
  accessModes:
    - ReadOnlyMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
  dataSource:
    kind: PersistentVolumeClaim
    name: gcnv-san-rwo
```

#### ReadWriteMany (RWX) — solo bloque en bruto

ReadWriteMany solo se admite cuando `volumeMode: Block`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-raw-rwx
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

##### Comportamiento del volumen de bloques

Los volúmenes de bloques se aprovisionan como LUN iSCSI y se presentan a los nodos de Kubernetes como dispositivos de bloque.

Volúmenes en bloque:

- Usa el protocolo iSCSI
- Soporte para sistema de archivos y presentación de bloques sin procesar
- Están adjuntos y gestionados por Trident
- Soporta múltiples modos de acceso a Kubernetes

#### Modos de acceso

Los volúmenes en bloque aprovisionados por Trident admiten los siguientes modos de acceso:

- `ReadWriteOnce` (RWO)
- `ReadOnlyMany` (ROX)
- `ReadWriteOncePod` (RWOP)
- `ReadWriteMany` (RWX), compatible solo cuando `volumeMode: Block`

#### volumeMode comportamiento

El `volumeMode` campo controla cómo se expone un volumen de bloque:

- `Filesystem` Trident formatea y monta el volumen.
- `Block` Trident conecta el dispositivo y lo expone como un dispositivo de bloque sin procesar.

#### Operaciones compatibles

Los volúmenes en bloque aprovisionados usando el controlador `google-cloud-netapp-volumes-san` admiten:

- Crear
- Borrar
- Clonar
- Snapshot
- Cambiar tamaño
- Importar

#### Comportamiento de la sobreaprovisionamiento de GiB extra

Google Cloud NetApp Volumes los volúmenes en bloque incluyen una sobrecarga interna de metadatos. Esta sobrecarga reduce el tamaño del dispositivo visible desde el kernel en comparación con la capacidad provisionada.

Las pruebas muestran:

- Aproximadamente 300 KiB de sobrecarga en la creación inicial
- Hasta aproximadamente 107 MiB de sobrecarga después de un redimensionamiento

Porque Google Cloud NetApp Volumes acepta solo asignaciones de GiB enteros, Trident se asegura de que el tamaño utilizable del dispositivo siempre cumpla o supere la solicitud de PVC:

- Redondeando el tamaño solicitado al siguiente GiB entero
- Agregando un búfer adicional de 1 GiB

Ejemplo:

- Solicitud de PVC: 100 GiB
- Tamaño aprovisionado en Google Cloud NetApp Volumes: 101 GiB

- Espacio útil visible para la aplicación: al menos 100 GiB

#### Ejemplos de pods

### Volumen de bloque montado en el sistema de archivos (RWO)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-rwo
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeMounts:
    - name: data
      mountPath: /mnt/data
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-rwo
```

### Dispositivo de bloque raw (RWX)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-raw-rwx
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeDevices:
    - name: data
      devicePath: /dev/xda
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-raw-rwx
```

#### Comportamiento de attach y mount

Para volúmenes SAN aprovisionados desde Google Cloud NetApp Volumes:

- Trident crea un Logical Unit Number (LUN) en un grupo de almacenamiento Flex Unified.
- Durante la publicación, Trident asigna el LUN a un grupo de hosts por nodo.
- Durante la puesta en escena del nodo, Trident:
  - Inicia sesión en el target iSCSI
  - Descubre el LUN
  - Configura multivía
- Si `volumeMode: Filesystem`, Trident formatea el dispositivo si es necesario y lo monta.
- Si `volumeMode: Block`, Trident conecta el dispositivo y lo expone directamente al pod sin formatear ni montar.

> Los volúmenes de bloques SAN no proporcionan bloqueo distribuido ni coordinación de escritura. Cuando un volumen de bloques es accedido por varios nodos (ReadWriteMany con `volumeMode: Block`), la aplicación o el sistema de archivos debe gestionar la concurrencia.

### Prepárate para configurar un backend de Google Cloud NetApp Volumes

Antes de que puedas configurar tu backend de Google Cloud NetApp Volumes, necesitas asegurarte de que se cumplan los siguientes requisitos.

#### Requisitos previos para volúmenes NFS o SMB

Si estás usando Google Cloud NetApp Volumes por primera vez o en una ubicación nueva, necesitas hacer una configuración inicial para configurar Google Cloud NetApp Volumes y crear un volumen NFS o SMB. Consulta "Antes de empezar".

Asegúrate de tener lo siguiente antes de configurar el backend de Google Cloud NetApp Volumes:

- Una cuenta de Google Cloud configurada con el servicio Google Cloud NetApp Volumes. Consulta "Google Cloud NetApp Volumes".
- Número de proyecto de tu cuenta de Google Cloud. Consulta "Identificación de proyectos".
- Una cuenta de servicio de Google Cloud con el rol de NetApp Volumes Admin (`roles/netapp.admin`. Consulta "Roles y permisos de Identity and Access Management".
- Archivo de clave API para tu cuenta GCNV. Consulta "Crear una clave de cuenta de servicio"
- Un grupo de almacenamiento. Consulta "Descripción general de los storage pools".

Para obtener más información sobre cómo configurar el acceso a Google Cloud NetApp Volumes, consulta "Configura el acceso a Google Cloud NetApp Volumes".

### Opciones y ejemplos de configuración del backend de Google Cloud NetApp Volumes

Conoce las opciones de configuración de backend para Google Cloud NetApp Volumes y revisa ejemplos de configuración.

#### Opciones de configuración del backend

Cada backend aprovisiona volúmenes en una única región de Google Cloud. Para crear volúmenes en otras regiones, puedes definir backends adicionales.

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `version` | | Siempre 1 |
| `storageDriverName` | Nombre del controlador de almacenamiento | El valor de `storageDriverName` debe especificarse como "google-cloud-netapp-volumes". |
| `backendName` | (Opcional) Nombre personalizado del backend de almacenamiento | Nombre del driver + "_" + parte de la clave API |
| `storagePools` | Parámetro opcional usado para especificar grupos de almacenamiento para la creación de volúmenes. | |
| `projectNumber` | Número de proyecto de la cuenta de Google Cloud. El valor se encuentra en la página de inicio del portal de Google Cloud. | |
| `location` | La ubicación de Google Cloud donde Trident crea volúmenes GCNV. Al crear clústeres de Kubernetes entre regiones, los volúmenes creados en un `location` se pueden usar en cargas de trabajo programadas en nodos de varias regiones de Google Cloud. El tráfico entre regiones tiene un costo adicional. | |
| `apiKey` | Clave API para la cuenta de servicio de Google Cloud con la `netapp.admin` función. Incluye el contenido en formato JSON del archivo de clave privada de una cuenta de servicio de Google Cloud (copiado textualmente en el archivo de configuración del backend). El `apiKey` debe incluir pares clave-valor para las siguientes claves: `type`, `project_id`, `client_email`, `client_id`, `auth_uri`, `token_uri`, `auth_provider_x509_cert_url` y `client_x509_cert_url`. | |
| `nfsMountOptions` | Control detallado de las opciones de montaje de NFS. | "nfsvers=3" |
| `limitVolumeSize` | Falla el aprovisionamiento si el tamaño del volumen solicitado es superior a este valor. | "" (no aplicado por defecto) |
| `serviceLevel` | El nivel de servicio de un pool de almacenamiento y sus volúmenes. Los valores son `flex`, `standard`, `premium` o `extreme`. | |
| `labels` | Conjunto de etiquetas arbitrarias con formato JSON para aplicar en volúmenes | "" |
| `network` | Red de Google Cloud utilizada para volúmenes GCNV. | |
| `debugTraceFlags` | Indicadores de depuración para utilizar cuando estés solucionando problemas. Ejemplo, `{"api":false, "method":true}`. No uses esto a menos que estés resolviendo problemas y necesites un volcado detallado del registro. | null |

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `nasType` | Configura la creación de volúmenes NFS o SMB. Las opciones son `nfs`, `smb` o null. Si lo configuras en null, se usarán volúmenes NFS por defecto. | `nfs` |
| `supportedTopologies` | Representa una lista de regiones y zonas compatibles con este backend. Para obtener más información, consulta "Usa la topología CSI". Por ejemplo: `supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a` | |

#### Opciones de aprovisionamiento de volumen

Puedes controlar el aprovisionamiento de volúmenes predeterminados en la sección `defaults` del archivo de configuración.

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `exportRule` | Las reglas de exportación para nuevos volúmenes. Debe ser una lista separada por comas de cualquier combinación de direcciones IPv4. | "0.0.0.0/0" |
| `snapshotDir` | Acceso al `.snapshot` directorio | `true`, `false` (el comportamiento por defecto puede variar. Establece explícitamente) "false" para NFSv3 |
| `snapshotReserve` | Porcentaje de volumen reservado para instantáneas | "" (acepta el valor predeterminado de 0) |
| `unixPermissions` | Los permisos unix de nuevos volúmenes (4 dígitos octales). | "" |

#### Configuraciones de ejemplo

Los siguientes ejemplos muestran configuraciones básicas que dejan la mayoría de los parámetros en sus valores predeterminados. Esta es la forma más fácil de definir un backend.

**Configuración mínima**

Esta es la configuración mínima absoluta del backend. Con esta configuración, Trident detecta todos tus grupos de almacenamiento delegados a Google Cloud NetApp Volumes en la ubicación configurada y coloca los nuevos volúmenes en uno de esos grupos aleatoriamente. Porque `nasType` se omite, la `nfs` configuración predeterminada se aplica y el backend aprovisionará volúmenes NFS.

Esta configuración es ideal cuando estás empezando a usar Google Cloud NetApp Volumes y probando cosas, pero en la práctica podrías necesitar proporcionar un alcance adicional para los volúmenes que aprovisionas.

Reemplaza `<id_value>` y `<key_value>` con las credenciales de tu cuenta de servicio.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

**Configuración para volúmenes SMB**

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

**Configuración con filtro StoragePools**

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  storagePools:
    - premium-pool1-europe-west6
    - premium-pool2-europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

**Configuración de virtual pool**

Esta configuración de backend define varios grupos virtuales en un solo archivo. Los grupos virtuales se definen en la `storage` sección. Son útiles cuando tienes varios grupos de almacenamiento que admiten diferentes niveles de servicio y quieres crear clases de almacenamiento en Kubernetes que los representen. Las etiquetas de los grupos virtuales se usan para diferenciarlos. Por ejemplo, en el ejemplo de abajo, la `performance` etiqueta y el `serviceLevel` tipo se usan para diferenciar los grupos virtuales.

También puedes establecer algunos valores predeterminados que se aplicarán a todos los grupos virtuales y sobrescribir esos valores predeterminados para grupos virtuales individuales. En el siguiente ejemplo, `snapshotReserve` y `exportRule` sirven como valores predeterminados para todos los grupos virtuales.

Para obtener más información, consulta "Pools virtuales".

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
```

```
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
        performance: standard
      serviceLevel: standard
```

**Identidad en la nube para GKE**

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

#### Configuración de topologías admitidas

Trident facilita el aprovisionamiento de volúmenes para cargas de trabajo según regiones y zonas de disponibilidad. El `supportedTopologies` bloque en esta configuración de backend se usa para proporcionar una lista de regiones y zonas por backend. Los valores de región y zona especificados aquí deben coincidir con los valores de región y zona de las etiquetas en cada nodo del clúster de Kubernetes. Estas regiones y zonas representan la lista de valores permitidos que se pueden proporcionar en una clase de almacenamiento. Para las clases de almacenamiento que contienen un subconjunto de las regiones y zonas proporcionadas en un backend, Trident crea volúmenes en la región y zona mencionadas. Para obtener más información, consulta "Usa la topología CSI".

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-a
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-b
```

#### ¿Qué sigue?

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

```
kubectl create -f <backend-file>
```

Para verificar que el backend se haya creado correctamente, ejecuta el siguiente comando:

```
kubectl get tridentbackendconfig

NAME                BACKEND NAME        BACKEND UUID
PHASE   STATUS
backend-tbc-gcnv    backend-tbc-gcnv    b2fd1ff9-b234-477e-88fd-713913294f65
Bound   Success
```

Si falla la creación del backend, algo anda mal con la configuración del backend. Puedes describir el backend usando el `kubectl get tridentbackendconfig <backend-name>` comando o ver los registros para determinar la causa ejecutando el siguiente comando:

```
tridentctl logs
```

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

#### Definiciones de clases de almacenamiento

La siguiente es una `StorageClass`definición básica que se refiere al backend de arriba.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

**Ejemplos de definiciones usando el `parameter.selector` campo:**

Usando `parameter.selector` puedes especificar para cada `StorageClass` el "pool virtual" que se utiliza para alojar un volumen. El volumen tendrá los aspectos definidos en el pool elegido.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=standard
  backendType: google-cloud-netapp-volumes
```

Para más detalles sobre las clases de almacenamiento, consulta "Crear una clase de almacenamiento".

#### Definiciones de ejemplo para volúmenes SMB

Usando `nasType`, `node-stage-secret-name` y `node-stage-secret-namespace`, puedes especificar un volumen SMB y proporcionar las credenciales necesarias de Active Directory. Cualquier usuario/contraseña de Active Directory, con o sin permisos, puede usarse para el secreto de la etapa del nodo.

**Configuración básica en el espacio de nombres predeterminado**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

**Usando diferentes secretos por espacio de nombres**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

**Usando diferentes secretos por volumen**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb`filtros para grupos que admiten volúmenes SMB. `nasType: nfs` o `nasType: null`filtros para grupos NFS.

**Ejemplo de definición de PVC**

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc
```

Para verificar si el PVC está enlazado, ejecuta el siguiente comando:

```
kubectl get pvc gcnv-nfs-pvc

NAME          STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS AGE
gcnv-nfs-pvc  Bound    pvc-b00f2414-e229-40e6-9b16-ee03eb79a213  100Gi
RWX            gcnv-nfs-sc  1m
```

**Configura la jerarquización automática para Google Cloud NetApp Volumes**

La clasificación automática se configura a través de los parámetros de backend de Trident y las anotaciones de PersistentVolumeClaim durante el aprovisionamiento de volúmenes. Puedes configurar la clasificación automática para Google Cloud NetApp Volumes usando Trident.

**Descripción general**

La jerarquización automática permite a Trident aprovisionar volúmenes que mueven automáticamente los datos inactivos de un nivel de rendimiento a un nivel de capacidad. Esto reduce el coste de almacenamiento mientras se preserva el rendimiento para los datos a los que se accede con frecuencia.

Trident aplica la configuración de auto-tiering solo en el momento de la creación del volumen. Los cambios posteriores al aprovisionamiento no son compatibles con Trident 26.02.

**Conceptos**

### Organización automática en niveles

La jerarquización automática mueve los datos a los que se accede con poca frecuencia de un nivel de rendimiento a un nivel de capacidad según los patrones de acceso. El movimiento de datos ocurre de forma asíncrona y no es inmediato.

### Política de tiering

La política de jerarquización determina si la jerarquización automática está habilitada para un volumen.

Se admiten las siguientes políticas: * `auto`: permite la jerarquización automática basada en patrones de acceso * `none`: desactiva la jerarquización automática

### Días de refrigeración

Los días de enfriamiento especifican el número mínimo de días que un bloque de datos debe permanecer inactivo antes de que sea elegible para la clasificación por niveles. Los días de enfriamiento se aplican solo cuando la política de clasificación por niveles se establece en `auto`.

#### Modelo de configuración

### Ámbitos de configuración

La jerarquización automática puede configurarse en varios ámbitos:

- **Ámbito del storage pool** se aplica a todos los volúmenes aprovisionados desde el pool.
- **Ámbito del volumen** Aplica a un solo volumen a través de las anotaciones de PersistentVolumeClaim.

Trident determina la configuración efectiva según dónde se defina cada ajuste.

### Precedencia de la configuración

Cuando se define la misma configuración en varios ámbitos, Trident aplica el siguiente orden de precedencia:

1. Anotaciones de PersistentVolumeClaim
2. Configuración del backend Trident
3. Valores predeterminados del pool de almacenamiento

Los ajustes definidos con mayor precedencia reemplazan los valores de nivel inferior.

#### Funcionalidad compatible en Trident 26.02

Trident 26.02 es compatible con las siguientes funciones de auto-tiering para Google Cloud NetApp Volumes:

- Habilitar o deshabilitar la clasificación automática durante el aprovisionamiento de volúmenes
- Definir una política de niveles en la configuración del backend de Trident
- Anular la política de niveles y los días de enfriamiento por volumen usando anotaciones de PVC
- Configurar los días de enfriamiento para los volúmenes con la jerarquización automática activada

#### Funcionalidad no soportada en Trident 26.02

Las siguientes operaciones no son compatibles:

- Modificar la configuración de auto-tiering después de crear el volumen
- Cambio de las políticas de niveles en volúmenes existentes usando actualizaciones de Kubernetes
- Aplicar la configuración de jerarquización automática fuera de los flujos de trabajo de aprovisionamiento gestionados por Trident

#### Parámetros de configuración del backend

Los siguientes parámetros controlan el comportamiento del auto-tiering cuando se definen en la configuración del backend de Trident:

| Parámetro | Requerido | Descripción |
|---|---|---|
| tieringPolicy | No | Política de tiering para volúmenes (`auto` o `none`) |
| tieringMinimumCoolingDays | No | Número de días inactivos antes de que los datos se transfieran de nivel (intervalo: 2–183, predeterminado: 31) |

#### Anulaciones a nivel de volumen usando anotaciones de PersistentVolumeClaim

### Anotaciones compatibles

Las anotaciones de PersistentVolumeClaim permiten anular por volumen la configuración de auto-tiering.

| Anotación | Descripción |
|---|---|
| `trident.netapp.io/tieringPolicy` | Anula la política de tiering del volumen |
| `trident.netapp.io/tieringMinimumCoolingDays` | Anula el valor de los días de cooling para el volumen |

### Ejemplo: PersistentVolumeClaim con anulaciones de auto-tiering

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-tiering-pvc
  annotations:
    trident.netapp.io/tieringPolicy: auto
    trident.netapp.io/tieringMinimumCoolingDays: "45"
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: google-cloud-netapp-volumes-auto-tiering
  resources:
    requests:
      storage: 500Gi
```

### Comportamiento del aprovisionamiento

- Los ajustes de clasificación automática se evalúan y aplican solo en el momento de la creación del volumen.
- Trident no reconcilia la configuración de tiering después del aprovisionamiento.
- Los días de enfriamiento se ignoran cuando la política de niveles se establece en `none`.

### Limitaciones de la plataforma

- La jerarquización automática solo es compatible con volúmenes NAS (NFS y SMB).
- Los volúmenes en bloque (iSCSI) no admiten auto-tiering.
- El grupo de almacenamiento de Google Cloud NetApp Volumes debe tener la auto-tiering activada en Google Cloud.

### Valores admitidos

- Rango válido para `tieringMinimumCoolingDays`: 2 a 183
- Valor predeterminado: 31

## Configura un backend de NetApp HCI o SolidFire

Aprende cómo crear y usar un backend de Element con tu instalación de Trident.

### Detalles del controlador Element

Trident proporciona el `solidfire-san` controlador de almacenamiento para comunicarse con el clúster. Los modos de acceso admitidos son: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

El `solidfire-san` controlador de almacenamiento admite los modos de volumen *file* y *block*. Para el `Filesystem` volumeMode, Trident crea un volumen y crea un sistema de archivos. El tipo de sistema de archivos se especifica mediante el StorageClass.

| Controlador | Protocolo | VolumeMode | Modos de acceso admitidos | Sistemas de archivos compatibles |
|---|---|---|---|---|
| `solidfire-san` | iSCSI | Bloque | RWO, ROX, RWX, RWOP | Sin sistema de archivos. Dispositivo de bloque sin formato. |
| `solidfire-san` | iSCSI | Sistema de archivos | RWO, RWOP | `xfs`, `ext3`, `ext4` |

### Antes de empezar

Necesitarás lo siguiente antes de crear un backend de Element.

- Un sistema de almacenamiento compatible que ejecuta Element software.
- Credenciales para un administrador o usuario inquilino de un clúster NetApp HCI/SolidFire que pueda administrar volúmenes.
- Todos tus nodos de trabajo de Kubernetes deben tener instaladas las herramientas iSCSI adecuadas. Consulta "Información de preparación del nodo worker".

### Opciones de configuración del backend

Consulta la siguiente tabla para ver las opciones de configuración del backend:

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `version` | | Siempre 1 |
| `storageDriverName` | Nombre del controlador de almacenamiento | Siempre "solidfire-san" |
| `backendName` | Nombre personalizado o el backend de almacenamiento | "solidfire_" + dirección IP de almacenamiento (iSCSI) |
| `Endpoint` | MVIP para el clúster SolidFire con credenciales de inquilino | |
| `SVIP` | Dirección IP y puerto de almacenamiento (iSCSI) | |
| `labels` | Conjunto de etiquetas arbitrarias con formato JSON para aplicar en volúmenes. | "" |
| `TenantName` | Nombre del tenant a usar (se crea si no se encuentra) | |
| `InitiatorIFace` | Restringe el tráfico iSCSI a una interfaz del host específica | "default" |
| `UseCHAP` | Usa CHAP para autenticar iSCSI. Trident usa CHAP. | verdadero |
| `AccessGroups` | Lista de ID de grupos de acceso para usar | Encuentra el ID de un grupo de acceso llamado "trident" |
| `Types` | Especificaciones de QoS | |
| `limitVolumeSize` | Falla el aprovisionamiento si el tamaño del volumen solicitado es superior a este valor | "" (no aplicado por defecto) |
| `debugTraceFlags` | Indicadores de depuración para utilizar cuando estés solucionando problemas. Por ejemplo, {"api":false, "method":true} | null |

**ADVERTENCIA** No uses `debugTraceFlags` a menos que estés solucionando problemas y necesites un volcado de registro detallado.

#### Ejemplo 1: configuración de backend para `solidfire-san` driver con tres tipos de volumen

Este ejemplo muestra un archivo backend que utiliza autenticación CHAP y modela tres tipos de volúmenes con garantías de QoS específicas. Lo más probable es que luego definas clases de almacenamiento para consumir cada uno de estos usando el `IOPS` parámetro de clase de almacenamiento.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
TenantName: <tenant>
labels:
  k8scluster: dev1
  backend: dev1-element-cluster
UseCHAP: true
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
```

#### Ejemplo 2: configuración de backend y clase de almacenamiento para `solidfire-san`driver con grupos virtuales

Este ejemplo muestra el archivo de definición de backend configurado con grupos virtuales junto con StorageClasses que hacen referencia a ellos.

Trident copia las etiquetas presentes en un pool de almacenamiento al LUN de almacenamiento backend durante el aprovisionamiento. Para mayor comodidad, los administradores de almacenamiento pueden definir etiquetas por pool virtual y agrupar volúmenes por etiqueta.

En el archivo de definición de backend de ejemplo que se muestra a continuación, se establecen valores predeterminados específicos para todos los grupos de almacenamiento, que establecen el `type` en Silver. Los grupos virtuales se definen en la sección `storage`. En este ejemplo, algunos grupos de almacenamiento establecen su propio tipo y otros anulan los valores predeterminados establecidos arriba.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
TenantName: <tenant>
UseCHAP: true
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
  - labels:
      performance: gold
      cost: "4"
    zone: us-east-1a
    type: Gold
  - labels:
      performance: silver
      cost: "3"
    zone: us-east-1b
    type: Silver
  - labels:
      performance: bronze
      cost: "2"
    zone: us-east-1c
    type: Bronze
  - labels:
      performance: silver
```

```
      cost: "1"
    zone: us-east-1d
```

Las siguientes definiciones de StorageClass se refieren a los grupos virtuales mencionados arriba. Usando el campo `parameters.selector`, cada StorageClass indica qué grupo(s) virtual(es) se pueden usar para alojar un volumen. El volumen tendrá los aspectos definidos en el grupo virtual elegido.

El primer StorageClass (`solidfire-gold-four`) se asignará al primer pool virtual. Este es el único pool que ofrece rendimiento gold con un `Volume Type QoS` de Gold. El último StorageClass (`solidfire-silver`) identifica cualquier pool de almacenamiento que ofrezca un rendimiento silver. Trident decidirá qué grupo virtual se selecciona y se asegurará de que se cumpla el requisito de almacenamiento.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold; cost=4
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=3
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze; cost=2
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
```

```
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=1
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
  fsType: ext4
```

#### Encuentra más información

- "Grupos de acceso a volúmenes"

## Controladores SAN de ONTAP

### Descripción general del controlador SAN de ONTAP

Conoce cómo configurar un backend de ONTAP con los controladores SAN de ONTAP y Cloud Volumes ONTAP.

#### Detalles del controlador SAN de ONTAP

Trident proporciona los siguientes controladores de almacenamiento SAN para comunicarse con el clúster ONTAP. Los modos de acceso admitidos son: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

| Controlador | Protocolo | volumeMode | Modos de acceso admitidos | Sistemas de archivos compatibles |
|---|---|---|---|---|
| `ontap-san` | iSCSI SCSI sobre FC | Bloque | RWO, ROX, RWX, RWOP | Sin sistema de archivos; dispositivo de bloque sin formato |
| `ontap-san` | iSCSI SCSI sobre FC | Sistema de archivos | RWO, RWOP<br><br>ROX y RWX no están disponibles en el modo de volumen de sistema de archivos. | `xfs`, `ext3`, `ext4` |

| Controlador | Protocolo | volumeMode | Modos de acceso admitidos | Sistemas de archivos compatibles |
|---|---|---|---|---|
| `ontap-san` | NVMe/TCP<br><br>Consulta Consideraciones adicionales para NVMe/TCP. | Bloque | RWO, ROX, RWX, RWOP | Sin sistema de archivos; dispositivo de bloque sin formato |
| `ontap-san` | NVMe/TCP<br><br>Consulta Consideraciones adicionales para NVMe/TCP. | Sistema de archivos | RWO, RWOP<br><br>ROX y RWX no están disponibles en el modo de volumen de sistema de archivos. | `xfs`, `ext3`, `ext4` |
| `ontap-san-economy` | iSCSI | Bloque | RWO, ROX, RWX, RWOP | Sin sistema de archivos; dispositivo de bloque sin formato |
| `ontap-san-economy` | iSCSI | Sistema de archivos | RWO, RWOP<br><br>ROX y RWX no están disponibles en el modo de volumen de sistema de archivos. | `xfs`, `ext3`, `ext4` |

**ADVERTENCIA**

- Usa `ontap-san-economy` solo si esperas que el recuento de uso de volúmenes persistentes sea mayor que "límites de volúmenes compatibles de ONTAP".
- Usa `ontap-nas-economy` solo si se espera que el recuento de uso de volúmenes persistentes sea mayor que "límites de volúmenes compatibles de ONTAP" y no se puede usar el controlador `ontap-san-economy`.
- No uses `ontap-nas-economy` si crees que vas a necesitar protección de datos, recuperación ante desastres o movilidad.
- NetApp no recomienda usar FlexVol autogrow en todos los controladores ONTAP, excepto ontap-san. Como solución alternativa, Trident admite el uso de snapshot reserve y escala los volúmenes FlexVol en consecuencia.

#### Permisos de usuario

Trident espera ejecutarse como administrador de ONTAP o SVM, normalmente usando el `admin`usuario del clúster o un `vsadmin`usuario de SVM, o un usuario con un nombre diferente que tenga el mismo rol. Para las implementaciones de Amazon FSx for NetApp ONTAP, Trident espera ejecutarse como administrador de ONTAP o SVM, usando el usuario

del clúster `fsxadmin o un `vsadmin`usuario de SVM, o un usuario con un nombre diferente que tenga el mismo rol. El `fsxadmin`usuario es un reemplazo limitado para el usuario administrador del clúster.

| | |
|---|---|
| **NOTA** | Si usas el parámetro `limitAggregateUsage`, se requieren permisos de administrador de clúster. Cuando usas Amazon FSx for NetApp ONTAP con Trident, el parámetro `limitAggregateUsage` no funcionará con las cuentas de usuario `vsadmin` y `fsxadmin`. La operación de configuración fallará si especificas este parámetro. |

Aunque es posible crear un rol más restrictivo dentro de ONTAP que un controlador Trident pueda usar, no lo recomendamos. La mayoría de las nuevas versiones de Trident llamarán a APIs adicionales que habría que tener en cuenta, lo que hace que las actualizaciones sean difíciles y propensas a errores.

#### Consideraciones adicionales para NVMe/TCP

Trident admite el protocolo non-volatile memory express (NVMe) usando el `ontap-san` driver, incluyendo:

- IPv6
- Instantáneas y clones de volúmenes NVMe
- Cambiar el tamaño de un volumen NVMe
- Importar un volumen NVMe que se creó fuera de Trident para que su ciclo de vida pueda ser gestionado por Trident
- Multipathing nativo de NVMe
- Apagado correcto o incorrecto de los nodos K8s (24.06)

Trident no admite:

- DH-HMAC-CHAP que es compatible de forma nativa con NVMe
- Mapeador de dispositivos (DM) multipathing
- Cifrado LUKS

| | |
|---|---|
| **NOTA** | NVMe solo es compatible con las API REST de ONTAP y no es compatible con ONTAPI (ZAPI). |

### Prepárate para configurar el backend con controladores SAN de ONTAP

## Entiende los requisitos y las opciones de autenticación para configurar un backend de ONTAP con controladores SAN de ONTAP.

#### Requisitos

Para todos los backends de ONTAP, Trident requiere que al menos un agregado esté asignado a la SVM.

| | |
|---|---|
| **NOTA** | "Sistemas ASA r2" se diferencian de otros sistemas ONTAP (ASA, AFF y FAS) en la implementación de su capa de almacenamiento. En los sistemas ASA r2, se utilizan zonas de disponibilidad de almacenamiento en lugar de agregados. Consulta el "esto" artículo de base de conocimientos sobre cómo asignar agregados a SVMs en sistemas ASA r2. |

Recuerda que también puedes ejecutar más de un controlador y crear clases de almacenamiento que apunten a uno u otro. Por ejemplo, podrías configurar una `san-dev` clase que use el `ontap-san` controlador y una

`san-default` clase que use el `ontap-san-economy` otro.

Todos tus nodos de trabajo de Kubernetes deben tener instaladas las herramientas iSCSI adecuadas. Consulta "Prepara el nodo trabajador" para más detalles.

#### Autentica el backend de ONTAP

Trident ofrece dos modos de autenticación para un backend ONTAP.

- Basado en credenciales: el nombre de usuario y la contraseña de un usuario de ONTAP con los permisos necesarios. Se recomienda usar un rol de inicio de sesión de seguridad predefinido, como `admin` o `vsadmin` para garantizar la máxima compatibilidad con las versiones de ONTAP.
- Basado en certificado: Trident también puede comunicarse con un clúster de ONTAP usando un certificado instalado en el backend. Aquí, la definición del backend debe contener valores codificados en Base64 del certificado del cliente, la clave y el certificado de la CA de confianza si se usa (recomendado).

Puedes actualizar los backends existentes para pasar de métodos basados en credenciales a métodos basados en certificados y viceversa. Sin embargo, solo se admite un método de autenticación a la vez. Para cambiar a otro método de autenticación, debes eliminar el método existente de la configuración del backend.

| | |
|---|---|
| **ADVERTENCIA** | Si intentas proporcionar **tanto credenciales como certificados**, la creación del backend fallará con un error que indica que se proporcionó más de un método de autenticación en el archivo de configuración. |

#### Habilita la autenticación basada en credenciales

Trident requiere las credenciales de un administrador con ámbito de SVM o de clúster para comunicarse con el backend de ONTAP. Se recomienda usar roles estándar predefinidos como `admin` o `vsadmin`. Esto garantiza la compatibilidad futura con versiones de ONTAP que podrían exponer APIs de funciones para futuras versiones de Trident. Se puede crear y usar un rol personalizado de inicio de sesión de seguridad con Trident, pero no se recomienda.

Un ejemplo de definición de backend se verá así:

**YAML**

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

**JSON**

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

Ten en cuenta que la definición del backend es el único lugar donde se almacenan las credenciales en texto plano. Después de crear el backend, los nombres de usuario y contraseñas se codifican con Base64 y se guardan como secretos de Kubernetes. La creación o actualización de un backend es el único paso que requiere conocimiento de las credenciales. Así que es una operación solo para el administrador, que debe realizar el administrador de Kubernetes o de almacenamiento.

### Habilita la autenticación basada en certificados

Los backends, tanto nuevos como existentes, pueden usar un certificado y comunicarse con el backend de ONTAP. Se requieren tres parámetros en la definición del backend.

- clientCertificate: valor codificado en Base64 del certificado de cliente.
- clientPrivateKey: Valor codificado en Base64 de la clave privada asociada.
- trustedCACertificate: Valor codificado en Base64 del certificado de CA de confianza. Si usas una CA de confianza, tienes que proporcionar este parámetro. Esto se puede ignorar si no usas una CA de confianza.

Un flujo de trabajo típico implica los siguientes pasos.

**Pasos**

1. Genera un certificado y una clave de cliente. Al generarlos, asigna el nombre común (CN) al usuario de ONTAP con el que te vas a autenticar.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. Agrega un certificado de CA de confianza al clúster de ONTAP. Esto puede que ya lo haya gestionado el administrador de almacenamiento. Ignora esto si no se usa ninguna CA de confianza.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Instala el certificado y la clave del cliente (del paso 1) en el clúster ONTAP.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

**NOTA** Después de ejecutar este comando, ONTAP solicita la introducción del certificado. Pega el contenido del `k8senv.pem` archivo generado en el paso 1, luego introduce `END` para completar la instalación.

4. Confirma que el rol de inicio de sesión de seguridad de ONTAP admite `cert` el método de autenticación.

```
security login create -user-or-group-name admin -application ontapi
-authentication-method cert
security login create -user-or-group-name admin -application http
-authentication-method cert
```

5. Prueba la autenticación usando el certificado generado. Reemplaza <ONTAP Management LIF> y <vserver name> con la IP de la LIF de administración y el nombre de la SVM.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Codifica el certificado, la clave y el certificado de CA confiable con Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Crea un backend usando los valores obtenidos en el paso anterior.

```
cat cert-backend.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"trustedCACertificate": "QNFinfO...SiqOyN",
"storagePrefix": "myPrefix_"
}

tridentctl create backend -f cert-backend.json -n trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       0 |
+------------+----------------+--------------------------------------
+--------+---------+
```

### Actualiza los métodos de autenticación o rota las credenciales

Puedes actualizar un backend existente para usar un método de autenticación diferente o para rotar sus credenciales. Esto funciona en ambos sentidos: los backends que usan nombre de usuario y contraseña pueden actualizarse para usar certificados; los backends que utilizan certificados pueden actualizarse para usar nombre de usuario y contraseña. Para hacer esto, debes eliminar el método de autenticación existente y agregar el nuevo método de autenticación. Luego usa el archivo backend.json actualizado que contiene los parámetros necesarios para ejecutar `tridentctl backend update`.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

| | |
|---|---|
| **NOTA** | Al rotar contraseñas, el administrador de almacenamiento debe primero actualizar la contraseña del usuario en ONTAP. Luego, se realiza una actualización del backend. Al rotar certificados, se pueden agregar varios certificados al usuario. Después, el backend se actualiza para usar el nuevo certificado, y luego se puede eliminar el certificado antiguo del clúster de ONTAP. |

Actualizar un backend no interrumpe el acceso a los volúmenes ya creados ni afecta las conexiones de volumen realizadas después. Una actualización correcta del backend indica que Trident puede comunicarse con el backend de ONTAP y gestionar futuras operaciones de volumen.

### Crear rol ONTAP personalizado para Trident

Puedes crear un rol de clúster de ONTAP con privilegios mínimos para que no tengas que usar el rol de admin de ONTAP para realizar operaciones en Trident. Cuando incluyes el nombre de usuario en una configuración de backend de Trident, Trident usa el rol de clúster de ONTAP que creaste para realizar las operaciones.

Consulta "Generador de roles personalizados de Trident" para más información sobre cómo crear roles personalizados de Trident.

**Usando ONTAP CLI**

1. Crea un nuevo rol usando el siguiente comando:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Crea un nombre de usuario para el usuario Trident:

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Asigna el rol al usuario:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

**Usando System Manager**

Realiza los siguientes pasos en ONTAP System Manager:

1. **Crea un rol personalizado**:
   a. Para crear un rol personalizado a nivel de cluster, selecciona **Cluster > Settings**.

      (O) Para crear un rol personalizado a nivel de SVM, selecciona **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**.

   b. Selecciona el icono de flecha (→) junto a **Users and Roles**.
   c. Selecciona **+Add** en **Roles**.
   d. Define las reglas para el rol y haz clic en **Guardar**.
2. **Asigna el rol al usuario Trident**: + Realiza los siguientes pasos en la página **Usuarios y roles**:
   a. Selecciona el icono Add **+** en **Usuarios**.
   b. Selecciona el nombre de usuario requerido y elige un rol en el menú desplegable de **Role**.
   c. Haz clic en **Guardar**.

Consulta las siguientes páginas para obtener más información:

- "Roles personalizados para la administración de ONTAP" o "Define roles personalizados"
- "Trabaja con roles y usuarios"

#### Autentica conexiones con CHAP bidireccional

Trident puede autenticar sesiones iSCSI con CHAP bidireccional para los `ontap-san` y `ontap-san-economy` controladores. Esto requiere habilitar la opción `useCHAP` en la definición de tu backend. Cuando se establece en `true`, Trident configura la seguridad del iniciador predeterminado de la SVM como CHAP bidireccional y establece el nombre de usuario y los secretos desde el archivo del backend. NetApp recomienda usar CHAP bidireccional para autenticar conexiones. Mira la siguiente configuración de ejemplo:

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

| ADVERTENCIA | El `useCHAP`parámetro es una opción booleana que solo se puede configurar una vez. Se establece en falso de forma predeterminada. Después de que lo configuras en verdadero, ya no puedes ponerlo en falso. |
|---|---|

Además de `useCHAP=true`, los `chapInitiatorSecret`, `chapTargetInitiatorSecret`, `chapTargetUsername` y `chapUsername` campos deben incluirse en la definición del backend. Los secretos se pueden cambiar después de crear un backend ejecutando `tridentctl update`.

#### Cómo funciona

Al establecer `useCHAP` en true, el administrador de almacenamiento le indica a Trident que configure CHAP en el backend de almacenamiento. Esto incluye lo siguiente:

- Configurar CHAP en la SVM:
  - Si el tipo de seguridad del iniciador predeterminado de la SVM es ninguno (establecido de manera predeterminada) **y** no hay LUN preexistentes en el volumen, Trident establecerá el tipo de seguridad predeterminado en `CHAP` y procederá a configurar el nombre de usuario y los secretos del iniciador y del destino CHAP.
  - Si la SVM contiene LUNs, Trident no habilitará CHAP en la SVM. Esto garantiza que el acceso a los LUNs que ya están presentes en la SVM no esté restringido.
- Configurar el iniciador CHAP y el nombre de usuario y los secretos del destino; estas opciones deben especificarse en la configuración del backend (como se muestra arriba).

Después de que se crea el backend, Trident crea un CRD correspondiente `tridentbackend` y almacena los secretos y nombres de usuario CHAP como secretos de Kubernetes. Todos los PV que Trident cree en este backend se montarán y conectarán mediante CHAP.

#### Rotar credenciales y actualizar backends

Puedes actualizar las credenciales CHAP actualizando los parámetros CHAP en el archivo `backend.json`. Esto requerirá actualizar los secretos CHAP y usar el comando `tridentctl update` para reflejar estos cambios.

| ADVERTENCIA | Al actualizar los secretos CHAP de un backend, debes usar `tridentctl` para actualizar el backend. No actualices las credenciales en el clúster de almacenamiento usando la CLI de ONTAP o el System Manager de ONTAP, ya que Trident no podrá detectar estos cambios. |
|---|---|

```
cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "password",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

./tridentctl update backend ontap_san_chap -f backend-san.json -n trident
+----------------+----------------+--------------------------------------
+--------+---------+
|   NAME         | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+----------------+----------------+--------------------------------------
+--------+---------+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |       7 |
+----------------+----------------+--------------------------------------
+--------+---------+
```

Las conexiones existentes no se verán afectadas; seguirán activas si Trident actualiza las credenciales en la SVM. Las nuevas conexiones usan las credenciales actualizadas y las conexiones existentes siguen activas. Desconectar y volver a conectar los PV antiguos hará que usen las credenciales actualizadas.

#### Opciones de configuración y ejemplos de ONTAP SAN

Aprende cómo crear y usar controladores ONTAP SAN con tu instalación de Trident. Esta sección ofrece ejemplos de configuración de backends y detalles para asignar backends a StorageClasses. "Sistemas ASA r2" se diferencian de otros sistemas ONTAP (ASA, AFF y FAS) en la implementación de su capa de almacenamiento. Estas variaciones afectan el uso de ciertos parámetros como se indica. "Conoce más sobre las diferencias entre los sistemas ASA r2 y otros sistemas ONTAP". En la configuración del backend de Trident, no necesitas especificar que tu sistema es ASA r2. Cuando seleccionas `ontap-`

`san` como `storageDriverName`, Trident detecta automáticamente el ASA r2 u otros sistemas ONTAP. Algunos parámetros de configuración del backend no aplican a los sistemas ASA r2, como se indica en la tabla de abajo.

**NOTA** Solo el controlador `ontap-san` (con protocolos iSCSI, NVMe/TCP y FC) es compatible con los sistemas ASA r2.

#### Opciones de configuración del backend

Consulta la siguiente tabla para ver las opciones de configuración del backend:

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `version` | | Siempre 1 |
| `storageDriverName` | Nombre del controlador de almacenamiento | `ontap-san` o `ontap-san-economy` |
| `backendName` | Nombre personalizado o el backend de almacenamiento | Nombre del driver + "_" + dataLIF |
| `managementLIF` | Dirección IP de un LIF de gestión de clúster o SVM.<br><br>Se puede especificar un nombre de dominio completo (FQDN).<br><br>Se puede configurar para usar direcciones IPv6 si Trident se instaló usando el flag de IPv6. Las direcciones IPv6 deben definirse entre corchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`.<br><br>Para una conmutación de sitios sin interrupciones de MetroCluster, consulta el Ejemplo de MetroCluster.<br><br>**NOTA** Si usas credenciales "vsadmin", `managementLIF` debe ser el de la SVM; si usas credenciales "admin", `managementLIF` debe ser el del clúster. | "10.0.0.1", "[2001:1234:abcd::fefe]" |
| `dataLIF` | Dirección IP del LIF de protocolo. Se puede configurar para usar direcciones IPv6 si Trident se instaló usando el flag de IPv6. Las direcciones IPv6 deben definirse entre corchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **No especifiques para iSCSI.** Trident usa "Mapa LUN selectivo de ONTAP" para descubrir los LIF de iSCSI necesarios para establecer una sesión multipath. Se genera una advertencia si `dataLIF` se define explícitamente. **Omite para MetroCluster.** Consulta la Ejemplo de MetroCluster. | Derivado por la SVM |

| Parámetro | Descripción | Predeterminado |
| --- | --- | --- |
| `svm` | Máquina virtual de almacenamiento a usar **Omitir para MetroCluster.** Consulta la Ejemplo de MetroCluster. | Se deriva si se especifica un SVM `managementLIF` |
| `useCHAP` | Usa CHAP para autenticar iSCSI para los controladores ONTAP SAN [parámetro booleano]. Establécelo en `true` para que Trident configure y use CHAP bidireccional como la autenticación predeterminada para el SVM dado en el backend. Consulta "Prepárate para configurar el backend con controladores SAN de ONTAP" para más detalles. **No es compatible con FCP o NVMe/TCP.** | `false` |
| `chapInitiatorSecret` | Secreto del iniciador de CHAP. Requerido si `useCHAP=true` | "" |
| `labels` | Conjunto de etiquetas arbitrarias con formato JSON para aplicar en volúmenes | "" |
| `chapTargetInitiatorSecret` | Secreto del iniciador de destino CHAP. Requerido si `useCHAP=true` | "" |
| `chapUsername` | Nombre de usuario de entrada. Requerido si `useCHAP=true` | "" |
| `chapTargetUsername` | Nombre de usuario de destino. Requerido si `useCHAP=true` | "" |
| `clientCertificate` | Valor codificado en Base64 del certificado del cliente. Usado para auth basada en certificados | "" |
| `clientPrivateKey` | Valor codificado en Base64 de la clave privada del cliente. Usado para auth basada en certificados | "" |
| `trustedCACertificate` | Valor codificado en Base64 del certificado de CA de confianza. Opcional. Se usa para la autenticación basada en certificados. | "" |
| `username` | Nombre de usuario necesario para comunicarte con el clúster ONTAP. Se utiliza para la autenticación basada en credenciales. Para la autenticación de Active Directory, mira "Autentica Trident en un SVM backend usando credenciales de Active Directory". | "" |
| `password` | Contraseña necesaria para comunicarte con el clúster ONTAP. Se utiliza para la autenticación basada en credenciales. Para la autenticación de Active Directory, mira "Autentica Trident en un SVM backend usando credenciales de Active Directory". | "" |
| `svm` | Máquina virtual de almacenamiento que vas a usar | Se deriva si se especifica un SVM `managementLIF` |

| Parámetro | Descripción | Predeterminado |
| --- | --- | --- |
| `storagePrefix` | Prefijo utilizado al aprovisionar nuevos volúmenes en la SVM. No se puede modificar después. Para actualizar este parámetro, tendrás que crear un nuevo backend. | `trident` |
| `aggregate` | Agregado para aprovisionamiento (opcional; si se configura, debe asignarse a la SVM). Para el `ontap-nas-flexgroup` driver, esta opción se ignora. Si no se asigna, cualquiera de los agregados disponibles se puede usar para aprovisionar un FlexGroup volumen.<br>**NOTA**<br>Cuando el agregado se actualiza en SVM, se actualiza automáticamente en Trident mediante el sondeo de SVM sin tener que reiniciar el Trident Controller. Cuando has configurado un agregado específico en Trident para aprovisionar volúmenes, si el agregado se renombra o se mueve fuera de la SVM, el backend pasará a estado fallido en Trident mientras sondea el agregado de la SVM. Debes cambiar el agregado por uno que esté presente en la SVM o eliminarlo por completo para que el backend vuelva a estar en línea.<br>**No especificar para sistemas ASA r2**. | "" |
| `limitAggregateUsage` | Falla el aprovisionamiento si el uso es superior a este porcentaje. Si estás usando un backend de Amazon FSx para NetApp ONTAP, no especifiques `limitAggregateUsage`. Los `fsxadmin` y `vsadmin` no tienen los permisos necesarios para recuperar el uso agregado y limitarlo usando Trident. **No especificar para sistemas ASA r2**. | "" (no aplicado por defecto) |
| `limitVolumeSize` | Falla el aprovisionamiento si el tamaño del volumen solicitado supera este valor. También restringe el tamaño máximo de los volúmenes que gestiona para LUNs. | "" (no aplicado por defecto) |
| `lunsPerFlexvol` | Máximo de LUNs por FlexVol, debe estar en el rango [50, 200] | `100` |
| `debugTraceFlags` | Indicadores de depuración para utilizar cuando estés solucionando problemas. Por ejemplo, {"api":false, "method":true} no lo uses a menos que estés solucionando problemas y necesites un volcado detallado del registro. | `null` |

| Parámetro | Descripción | Predeterminado |
| --- | --- | --- |
| `useREST` | Parámetro booleano para usar las ONTAP REST APIs.<br><br>`useREST`Cuando se establece en `true`, Trident usa las ONTAP REST APIs para comunicarse con el backend; cuando se establece en `false`, Trident usa llamadas ONTAPI (ZAPI) para comunicarse con el backend. Esta función requiere ONTAP 9.11.1 y versiones posteriores. Además, el rol de inicio de sesión de ONTAP utilizado debe tener acceso a la aplicación `ontapi`. Esto se cumple con los roles predefinidos `vsadmin` y `cluster-admin`. A partir de la versión Trident 24.06 y ONTAP 9.15.1 o posteriores, `useREST`está configurado en `true`de forma predeterminada; cambia `useREST` a `false` para usar llamadas ONTAPI (ZAPI).<br><br>`useREST` está totalmente calificado para NVMe/TCP.<br><br>**NOTA** NVMe solo es compatible con las API REST de ONTAP y no es compatible con ONTAPI (ZAPI).<br><br>**Si se especifica, siempre configúralo en `true` para sistemas ASA r2**. | `true` para ONTAP 9.15.1 o posterior, de lo contrario `false`. |
| `sanType` | Usa para seleccionar `iscsi` para iSCSI, `nvme` para NVMe/TCP o `fcp` para SCSI sobre Fibre Channel (FC). | `iscsi` si está en blanco |

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `formatOptions` | Usa `formatOptions` para especificar argumentos de la línea de comandos para el comando `mkfs`, que se aplicarán cada vez que se formatee un volumen. Esto te permite formatear el volumen según tus preferencias. Asegúrate de especificar formatOptions similar a las opciones del comando mkfs, excluyendo la ruta del dispositivo. Ejemplo: "-E nodiscard"<br><br>**Compatible con `ontap-san` y `ontap-san-economy` controladores con el protocolo iSCSI. Además, compatible con sistemas ASA r2 cuando usas los protocolos iSCSI y NVMe/TCP.** | |
| `limitVolumePoolSize` | Tamaño máximo solicitable de FlexVol al utilizar LUNs en el backend de ontap-san-economy. | "" (no aplicado por defecto) |
| `denyNewVolumePools` | Restringe `ontap-san-economy` a los backends crear nuevos volúmenes FlexVol para contener sus LUN. Solo se usan FlexVols preexistentes para aprovisionar nuevos PV. | |

#### Recomendaciones para usar formatOptions

Trident recomienda las siguientes opciones para acelerar el proceso de formateo:

- **-E nodiscard (ext3, ext4):** No intentes descartar bloques al momento de ejecutar mkfs (descartar bloques inicialmente es útil en dispositivos de estado sólido y almacenamiento disperso o Thin-Provisioning). Esto reemplaza la opción obsoleta "-K" y es aplicable a los sistemas de archivos ext3 y ext4.
- **-K (xfs):** No intentes descartar bloques durante la ejecución de mkfs. Esta opción es aplicable al sistema de archivos xfs.

#### Autentica Trident en un SVM backend usando credenciales de Active Directory

Puedes configurar Trident para que se autentique en una SVM de backend usando credenciales de Active Directory (AD). Antes de que una cuenta de AD pueda acceder a la SVM, tienes que configurar el acceso del controlador de dominio de AD al clúster o a la SVM. Para la administración del clúster con una cuenta de AD, tienes que crear un domain tunnel. Consulta "Configura el acceso al controlador de dominio de Active Directory en ONTAP" para más detalles.

**pasos**

1. Configura los ajustes de Domain Name System (DNS) para un SVM de backend:

   ```
   vserver services dns create -vserver <svm_name> -dns-servers
   <dns_server_ip1>,<dns_server_ip2>
   ```

2. Ejecuta el siguiente comando para crear una cuenta de equipo para la SVM en Active Directory:

   ```
   vserver active-directory create -vserver DataSVM -account-name ADSERVER1
   -domain demo.netapp.com
   ```

3. Usa este comando para crear un usuario o grupo de AD para administrar el clúster o SVM

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. En el archivo de configuración del backend de Trident, establece los parámetros `username` y `password` en el nombre de usuario o grupo de AD y la contraseña, respectivamente.

#### Opciones de configuración de backend para aprovisionar volúmenes

Puedes controlar el aprovisionamiento predeterminado usando estas opciones en la `defaults` sección de la configuración. Por ejemplo, mira los ejemplos de configuración abajo.

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `spaceAllocation` | Asignación de espacio para LUNs | "verdadero" **Si se especifica, configúralo en `true` para sistemas ASA r2**. |
| `spaceReserve` | Modo de reserva de espacio; "ninguno" (thin) o "volumen" (thick). **Configúralo en `none` sistemas ASA r2**. | "none" |
| `snapshotPolicy` | Política de SnapVault que se va a usar. **Establece en `none` para sistemas ASA r2**. | "none" |
| `qosPolicy` | Grupo de políticas de QoS para asignar a los volúmenes creados. Elige uno de qosPolicy o adaptiveQosPolicy por pool de almacenamiento/backend. Usar grupos de políticas de QoS con Trident requiere ONTAP 9.8 o una versión posterior. Deberías usar un grupo de políticas de QoS no compartido y asegurarte de que el grupo de políticas se aplique a cada componente individualmente. Un grupo de políticas de QoS compartido impone el límite máximo para el rendimiento total de todas las cargas de trabajo. | "" |
| `adaptiveQosPolicy` | Grupo de políticas de QoS adaptativo para asignar a los volúmenes creados. Elige uno de qosPolicy o adaptiveQosPolicy por cada pool de almacenamiento/backend | "" |
| `snapshotReserve` | Porcentaje de volumen reservado para instantáneas. **No especificar para sistemas ASA r2**. | "0" si `snapshotPolicy` es "none", de lo contrario "" |
| `splitOnClone` | Divide un clon de su padre al momento de su creación | "false" |
| `encryption` | Habilita NetApp Volume Encryption (NVE) en el nuevo volumen; el valor predeterminado es `false`. NVE debe tener licencia y estar habilitado en el clúster para usar esta opción. Si NAE está habilitado en el backend, cualquier volumen aprovisionado en Trident tendrá NAE habilitado. Para más información, consulta: "Cómo funciona Trident con NVE y NAE". | "falso" **Si se especifica, configúralo en `true` para sistemas ASA r2**. |

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `luksEncryption` | Activa el cifrado LUKS. Consulta "Usa Linux Unified Key Setup (LUKS)". | "" **Establécelo en `false` para sistemas ASA r2.** |
| `tieringPolicy` | Política de niveles para usar "none" **No especificar para sistemas ASA r2.** | |
| `nameTemplate` | Plantilla para crear nombres de volúmenes personalizados. | "" |

#### Ejemplos de aprovisionamiento de volumen

Aquí tienes un ejemplo con valores predeterminados definidos:

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

**NOTA** Para todos los volúmenes creados usando el `ontap-san` driver, Trident añade un 10 por ciento extra de capacidad al FlexVol para alojar los metadatos del LUN. El LUN se aprovisionará con el tamaño exacto que el usuario solicite en la PVC. Trident añade un 10 por ciento al FlexVol (se muestra como tamaño disponible en ONTAP). Ahora los usuarios obtendrán la cantidad de capacidad utilizable que solicitaron. Este cambio también evita que los LUN se vuelvan de solo lectura a menos que se utilice completamente el espacio disponible. Esto no aplica a ontap-san-economy.

Para los backends que definen `snapshotReserve`, Trident calcula el tamaño de los volúmenes de la siguiente manera:

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
percentage) / 100)] * 1.1
```

El 1.1 es el 10 % adicional que Trident añade a la FlexVol para alojar los metadatos del LUN. Para `snapshotReserve` = 5 %, y una solicitud de PVC de 5 GiB, el tamaño total del volumen es 5.79 GiB y el tamaño disponible es 5.5 GiB. El `volume show` comando debería mostrar resultados similares a este ejemplo:


Actualmente, cambiar el tamaño es la única manera de usar el nuevo cálculo para un volumen existente.

#### Ejemplos de configuración mínima

Los siguientes ejemplos muestran configuraciones básicas que dejan la mayoría de los parámetros en sus valores predeterminados. Esta es la forma más fácil de definir un backend.

| NOTA | Si estás usando Amazon FSx en NetApp ONTAP con Trident, NetApp recomienda que especifiques nombres DNS para las LIFs en vez de direcciones IP. |
|---|---|

#### Ejemplo de ONTAP SAN

Esta es una configuración básica usando el `ontap-san` driver.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

#### Ejemplo de MetroCluster

Puedes configurar el backend para evitar tener que actualizar manualmente la definición del backend después de la conmutación de sitios y la conmutación de vuelta durante "Replicación y recuperación de SVM".

Para una conmutación de sitios y reversión sin problemas, especifica el SVM usando `managementLIF` y omite los parámetros `svm`. Por ejemplo:

```
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

#### Ejemplo de ONTAP SAN economy

```
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
username: vsadmin
password: <password>
```

**Ejemplo de autenticación basada en certificados**

En este ejemplo de configuración básica `clientCertificate`, `clientPrivateKey` y `trustedCACertificate` (opcional, si usas una CA confiable) se completan en `backend.json` y toman los valores codificados en base64 del certificado del cliente, la clave privada y el certificado de CA confiable, respectivamente.

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

### Ejemplos de CHAP bidireccional

Estos ejemplos crean un backend con `useCHAP` configurado en `true`.

#### Ejemplo de ONTAP SAN CHAP

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

#### Ejemplo de ONTAP SAN economy CHAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

#### Ejemplo de NVMe/TCP

Debes tener una SVM configurada con NVMe en tu backend de ONTAP. Esta es una configuración básica de backend para NVMe/TCP.

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

#### Ejemplo de SCSI sobre FC (FCP)

Debes tener una SVM configurada con FC en tu backend de ONTAP. Esta es una configuración básica de backend para FC.

```
---
version: 1
backendName: fcp-backend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_fc
username: vsadmin
password: password
sanType: fcp
useREST: true
```

**Ejemplo de configuración de backend con nameTemplate**

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

**formatOptions ejemplo para el controlador ontap-san-economy**

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: ""
svm: svm1
username: ""
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: -E nodiscard
```

#### Ejemplos de backends con pools virtuales

En estos archivos de definición de backend de ejemplo, se establecen valores predeterminados específicos para todos los grupos de almacenamiento, como `spaceReserve` en ninguno, `spaceAllocation` en falso y `encryption` en falso. Los grupos virtuales se definen en la sección de almacenamiento.

Trident establece las etiquetas de aprovisionamiento en el campo "Comentarios". Los comentarios se establecen en el volumen FlexVol. Trident copia todas las etiquetas presentes en un pool virtual al volumen de almacenamiento durante el aprovisionamiento. Para mayor comodidad, los administradores de almacenamiento pueden definir etiquetas por pool virtual y agrupar volúmenes por etiqueta.

En estos ejemplos, algunos de los pools de almacenamiento establecen sus propios `spaceReserve`, `spaceAllocation` y `encryption` valores, y algunos pools anulan los valores predeterminados.

### Ejemplo de ONTAP SAN

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "40000"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
      adaptiveQosPolicy: adaptive-extreme
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
      qosPolicy: premium
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

**Ejemplo de ONTAP SAN economy**

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
labels:
  store: san_economy_store
region: us_east_1
storage:
  - labels:
      app: oracledb
      cost: "30"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
  - labels:
      app: postgresdb
      cost: "20"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
  - labels:
      app: mysqldb
      cost: "10"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
  - labels:
      department: legal
      creditpoints: "5000"
    zone: us_east_1c
```

```
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

**Ejemplo de NVMe/TCP**

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: "false"
  encryption: "true"
storage:
  - labels:
      app: testApp
      cost: "20"
    defaults:
      spaceAllocation: "false"
      encryption: "false"
```

#### Asigna backends a StorageClasses

Las siguientes definiciones de StorageClass se refieren a Ejemplos de backends con pools virtuales. Usando el campo `parameters.selector`, cada StorageClass indica qué grupos virtuales pueden usarse para alojar un volumen. El volumen tendrá los aspectos definidos en el grupo virtual elegido.

- El `protection-gold` StorageClass se asignará al primer pool virtual en el `ontap-san` backend. Este es el único pool que ofrece protección de nivel oro.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- El `protection-not-gold` StorageClass se asignará al segundo y tercer pool virtual en `ontap-san` backend. Estos son los únicos pools que ofrecen un nivel de protección distinto al oro.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- El `app-mysqldb` StorageClass se asignará al tercer pool virtual en `ontap-san-economy` backend. Este es el único pool que ofrece configuración de pool de almacenamiento para la app tipo mysqldb.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- El `protection-silver-creditpoints-20k` StorageClass se asignará al segundo pool virtual en `ontap-san` backend. Este es el único pool que ofrece protección de nivel plata y 20000 puntos de crédito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- El `creditpoints-5k` StorageClass se asignará al tercer grupo virtual en el `ontap-san` backend y al cuarto grupo virtual en el `ontap-san-economy` backend. Estas son las únicas ofertas de grupos con 5000 puntos de crédito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

- El `my-test-app-sc` StorageClass se asignará al `testAPP` grupo virtual en el `ontap-san` driver con `sanType: nvme`. Este es el único grupo que ofrece `testApp`.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"
```

Trident decidirá qué grupo virtual se selecciona y se asegurará de que se cumpla el requisito de almacenamiento.

## Controladores NAS de ONTAP

### Descripción general del controlador NAS de ONTAP

Conoce cómo configurar un backend de ONTAP con los controladores NAS de ONTAP y Cloud Volumes ONTAP.

#### Detalles del controlador NAS de ONTAP

Trident proporciona los siguientes controladores de almacenamiento NAS para comunicarse con el clúster ONTAP. Los modos de acceso admitidos son: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

| Controlador | Protocolo | volumeMode | Modos de acceso admitidos | Sistemas de archivos compatibles |
|---|---|---|---|---|
| `ontap-nas` | NFS SMB | Sistema de archivos | RWO, ROX, RWX, RWOP | "", `nfs`, `smb` |

| Controlador | Protocolo | volumeMode | Modos de acceso admitidos | Sistemas de archivos compatibles |
|---|---|---|---|---|
| `ontap-nas-economy` | NFS SMB | Sistema de archivos | RWO, ROX, RWX, RWOP | "", `nfs`, `smb` |
| `ontap-nas-flexgroup` | NFS SMB | Sistema de archivos | RWO, ROX, RWX, RWOP | "", `nfs`, `smb` |

**ADVERTENCIA**

- Usa `ontap-san-economy` solo si esperas que el recuento de uso de volúmenes persistentes sea mayor que "límites de volúmenes compatibles de ONTAP".
- Usa `ontap-nas-economy` solo si se espera que el recuento de uso de volúmenes persistentes sea mayor que "límites de volúmenes compatibles de ONTAP" y no se puede usar el controlador `ontap-san-economy`.
- No uses `ontap-nas-economy` si crees que vas a necesitar protección de datos, recuperación ante desastres o movilidad.
- NetApp no recomienda usar FlexVol autogrow en todos los controladores ONTAP, excepto ontap-san. Como solución alternativa, Trident admite el uso de snapshot reserve y escala los volúmenes FlexVol en consecuencia.

#### Permisos de usuario

Trident espera ejecutarse como administrador de ONTAP o SVM, normalmente usando el `admin`usuario del clúster o un `vsadmin`usuario de SVM, o un usuario con un nombre diferente que tenga el mismo rol.

Para las implementaciones de Amazon FSx for NetApp ONTAP, Trident espera ejecutarse como administrador de ONTAP o SVM, usando el usuario del clúster `fsxadmin` o un `vsadmin`usuario de SVM, o un usuario con un nombre diferente que tenga el mismo rol. El `fsxadmin`usuario es un reemplazo limitado para el usuario administrador del clúster.

**NOTA**

Si usas el parámetro `limitAggregateUsage`, se requieren permisos de administrador de clúster. Cuando usas Amazon FSx for NetApp ONTAP con Trident, el parámetro `limitAggregateUsage` no funcionará con las cuentas de usuario `vsadmin` y `fsxadmin`. La operación de configuración fallará si especificas este parámetro.

Aunque es posible crear un rol más restrictivo dentro de ONTAP que un controlador Trident pueda usar, no lo recomendamos. La mayoría de las nuevas versiones de Trident llamarán a APIs adicionales que habría que tener en cuenta, lo que hace que las actualizaciones sean difíciles y propensas a errores.

### Prepárate para configurar un backend con controladores ONTAP NAS

Entiende los requisitos, las opciones de autenticación y las políticas de exportación para configurar un backend ONTAP con drivers ONTAP NAS. A partir de la versión 25.10, NetApp Trident es compatible con "Sistema de almacenamiento NetApp AFX". NetApp AFX storage systems difieren de otros sistemas ONTAP (ASA, AFF y FAS) en la implementación de su capa de almacenamiento. En la configuración del backend de Trident, no necesitas especificar que tu sistema es AFX. Cuando seleccionas `ontap-`

`nas` como `storageDriverName`, Trident detecta automáticamente los sistemas AFX.

| NOTA | Sólo el `ontap-nas` controlador (con protocolo NFS) es compatible con los sistemas AFX; el protocolo SMB no es compatible. |
|---|---|

#### Requisitos

- Para todos los backends de ONTAP, Trident requiere que al menos un agregado esté asignado a la SVM.
- Puedes ejecutar más de un controlador y crear clases de almacenamiento que apunten a uno u otro. Por ejemplo, podrías configurar una clase Gold que use el `ontap-nas` controlador y una clase Bronze que use el `ontap-nas-economy` otro.
- Todos tus nodos worker de Kubernetes deben tener instaladas las herramientas NFS adecuadas. Consulta "aquí" para más detalles.
- Trident solo admite volúmenes SMB montados en pods que se ejecutan en nodos Windows. Consulta Prepárate para aprovisionar volúmenes SMB para más detalles.

#### Autentica el backend de ONTAP

Trident ofrece dos modos de autenticación para un backend ONTAP.

- Basado en credenciales: este modo requiere permisos suficientes en el backend de ONTAP. Se recomienda usar una cuenta asociada a un rol de inicio de sesión de seguridad predefinido, como `admin` o `vsadmin` para garantizar la máxima compatibilidad con las versiones de ONTAP.
- Basado en certificado: este modo requiere un certificado instalado en el backend para que Trident se comunique con un clúster de ONTAP. Aquí, la definición del backend debe contener valores codificados en Base64 del certificado del cliente, la clave y el certificado de la CA de confianza si se usa (recomendado).

Puedes actualizar los backends existentes para pasar de métodos basados en credenciales a métodos basados en certificados y viceversa. Sin embargo, solo se admite un método de autenticación a la vez. Para cambiar a otro método de autenticación, debes eliminar el método existente de la configuración del backend.

| ADVERTENCIA | Si intentas proporcionar **tanto credenciales como certificados**, la creación del backend fallará con un error que indica que se proporcionó más de un método de autenticación en el archivo de configuración. |
|---|---|

#### Habilita la autenticación basada en credenciales

Trident requiere las credenciales de un administrador con ámbito de SVM o de clúster para comunicarse con el backend de ONTAP. Se recomienda usar roles estándar predefinidos como `admin` o `vsadmin`. Esto garantiza la compatibilidad futura con versiones de ONTAP que podrían exponer APIs de funciones para futuras versiones de Trident. Se puede crear y usar un rol personalizado de inicio de sesión de seguridad con Trident, pero no se recomienda.

Un ejemplo de definición de backend se verá así:

**YAML**

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

**JSON**

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
        "name": "secret-backend-creds"
    }
}
```

Ten en cuenta que la definición del backend es el único lugar donde se almacenan las credenciales en texto plano. Después de crear el backend, los nombres de usuario y contraseñas se codifican con Base64 y se guardan como secretos de Kubernetes. La creación o actualización de un backend es el único paso que requiere conocimiento de las credenciales. Así que es una operación solo para el administrador, que debe realizar el administrador de Kubernetes o de almacenamiento.

### Habilita la autenticación basada en certificados

Los backends, tanto nuevos como existentes, pueden usar un certificado y comunicarse con el backend de ONTAP. Se requieren tres parámetros en la definición del backend.

- clientCertificate: valor codificado en Base64 del certificado de cliente.
- clientPrivateKey: Valor codificado en Base64 de la clave privada asociada.
- trustedCACertificate: Valor codificado en Base64 del certificado de CA de confianza. Si usas una CA de confianza, tienes que proporcionar este parámetro. Esto se puede ignorar si no usas una CA de confianza.

Un flujo de trabajo típico implica los siguientes pasos.

**Pasos**

1. Genera un certificado y una clave de cliente. Al generarlos, asigna el nombre común (CN) al usuario de

ONTAP con el que te vas a autenticar.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
```

2. Agrega un certificado de CA de confianza al clúster de ONTAP. Esto puede que ya lo haya gestionado el administrador de almacenamiento. Ignora esto si no se usa ninguna CA de confianza.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Instala el certificado y la clave del cliente (del paso 1) en el clúster ONTAP.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. Confirma que el rol de inicio de sesión de seguridad de ONTAP admite `cert` el método de autenticación.

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

5. Prueba la autenticación usando el certificado generado. Reemplaza <ONTAP Management LIF> y <vserver name> con la IP de la LIF de administración y el nombre de la SVM. Debes asegurarte de que el LIF tenga su política de servicio configurada en `default-data-management`.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Codifica el certificado, la clave y el certificado de CA confiable con Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Crea un backend usando los valores obtenidos en el paso anterior.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

#### Actualiza los métodos de autenticación o rota las credenciales

Puedes actualizar un backend existente para usar un método de autenticación diferente o para rotar sus credenciales. Esto funciona en ambos sentidos: los backends que usan nombre de usuario y contraseña pueden actualizarse para usar certificados; los backends que utilizan certificados pueden actualizarse para usar nombre de usuario y contraseña. Para hacer esto, debes eliminar el método de autenticación existente y agregar el nuevo método de autenticación. Luego usa el archivo backend.json actualizado que contiene los parámetros necesarios para ejecutar `tridentctl update backend`.

```
cat cert-backend-updated.json
```

```
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}
```

```
#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

**NOTA** Al rotar contraseñas, el administrador de almacenamiento debe primero actualizar la contraseña del usuario en ONTAP. Luego, se realiza una actualización del backend. Al rotar certificados, se pueden agregar varios certificados al usuario. Después, el backend se actualiza para usar el nuevo certificado, y luego se puede eliminar el certificado antiguo del clúster de ONTAP.

Actualizar un backend no interrumpe el acceso a los volúmenes ya creados ni afecta las conexiones de volumen realizadas después. Una actualización correcta del backend indica que Trident puede comunicarse con el backend de ONTAP y gestionar futuras operaciones de volumen.

### Crear rol ONTAP personalizado para Trident

Puedes crear un rol de clúster de ONTAP con privilegios mínimos para que no tengas que usar el rol de admin de ONTAP para realizar operaciones en Trident. Cuando incluyes el nombre de usuario en una configuración de backend de Trident, Trident usa el rol de clúster de ONTAP que creaste para realizar las operaciones.

Consulta "Generador de roles personalizados de Trident" para más información sobre cómo crear roles personalizados de Trident.

**Usando ONTAP CLI**

1. Crea un nuevo rol usando el siguiente comando:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Crea un nombre de usuario para el usuario Trident:

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Asigna el rol al usuario:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

**Usando System Manager**

Realiza los siguientes pasos en ONTAP System Manager:

1. **Crea un rol personalizado**:
   a. Para crear un rol personalizado a nivel de cluster, selecciona **Cluster > Settings**.

      (O) Para crear un rol personalizado a nivel de SVM, selecciona **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**.

   b. Selecciona el icono de flecha (→) junto a **Users and Roles**.
   c. Selecciona **+Add** en **Roles**.
   d. Define las reglas para el rol y haz clic en **Guardar**.
2. **Asigna el rol al usuario Trident**: + Realiza los siguientes pasos en la página **Usuarios y roles**:
   a. Selecciona el icono Add **+** en **Usuarios**.
   b. Selecciona el nombre de usuario requerido y elige un rol en el menú desplegable de **Role**.
   c. Haz clic en **Guardar**.

Consulta las siguientes páginas para obtener más información:

- "Roles personalizados para la administración de ONTAP" o "Define roles personalizados"
- "Trabaja con roles y usuarios"

#### Gestiona las políticas de exportación de NFS

Trident utiliza políticas de exportación NFS para controlar el acceso a los volúmenes que aprovisiona.

Trident ofrece dos opciones cuando trabajas con políticas de exportación:

- Trident puede gestionar dinámicamente la política de exportación por sí mismo; en este modo de funcionamiento, el administrador de almacenamiento especifica una lista de bloques CIDR que

representan direcciones IP admisibles. Trident añade automáticamente a la política de exportación las direcciones IP de nodo aplicables que estén dentro de estos rangos en el momento de la publicación. Como alternativa, cuando no se especifican CIDR, se añadirán a la política de exportación todas las direcciones IP de unidifusión de ámbito global que se encuentren en el nodo al que se está publicando el volumen.

- Los administradores de almacenamiento pueden crear una política de exportación y agregar reglas manualmente. Trident utiliza la política de exportación predeterminada a menos que se especifique un nombre de política de exportación diferente en la configuración.

#### Gestiona dinámicamente las políticas de exportación

Trident ofrece la capacidad de gestionar dinámicamente las políticas de exportación para los backends de ONTAP. Esto le da al administrador de almacenamiento la posibilidad de especificar un espacio de direcciones permitido para las direcciones IP de los nodos de trabajo, en vez de definir reglas explícitas manualmente. Esto simplifica mucho la gestión de las políticas de exportación; las modificaciones en la política de exportación ya no requieren intervención manual en el clúster de almacenamiento. Además, esto ayuda a restringir el acceso al clúster de almacenamiento solo a los nodos de trabajo que están montando volúmenes y tienen direcciones IP dentro del rango especificado, lo que permite una gestión automatizada y detallada.

**NOTA** No uses Network Address Translation (NAT) cuando uses políticas de exportación dinámicas. Con NAT, el controlador de almacenamiento ve la dirección NAT del frontend y no la dirección IP real del host, así que se denegará el acceso cuando no se encuentre coincidencia en las reglas de exportación.

#### Ejemplo

Hay dos opciones de configuración que deben utilizarse. Aquí tienes un ejemplo de definición de backend:

```
---
version: 1
storageDriverName: ontap-nas-economy
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
  - 192.168.0.0/24
autoExportPolicy: true
```

**NOTA** Al usar esta función, debes asegurarte de que la unión raíz de tu SVM tenga una política de exportación previamente creada con una regla de exportación que permita el bloque CIDR del nodo (como la política de exportación predeterminada). Sigue siempre la práctica recomendada por NetApp de dedicar una SVM para Trident.

Aquí tienes una explicación de cómo funciona esta función usando el ejemplo de arriba:

- `autoExportPolicy` se establece en `true`. Esto indica que Trident crea una política de exportación para cada volumen aprovisionado con este backend para la `svm1` SVM y gestiona la adición y eliminación de

reglas usando bloques de direcciones `autoexportCIDRs`. Hasta que un volumen se conecta a un nodo, el volumen utiliza una política de exportación vacía sin reglas para evitar el acceso no deseado a ese volumen. Cuando un volumen se publica en un nodo, Trident crea una política de exportación con el mismo nombre que el qtree subyacente que contiene la dirección IP del nodo dentro del bloque CIDR especificado. Estas direcciones IP también se añadirán a la política de exportación utilizada por el volumen principal FlexVol.

  - Por ejemplo:
    - UUID de backend 403b5326-8482-40db-96d0-d83fb3f4daec
    - `autoExportPolicy` establecer en `true`
    - prefijo de almacenamiento `trident`
    - PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
    - El qtree llamado trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c crea una política de exportación para el FlexVol llamado `trident-403b5326-8482-40db96d0-d83fb3f4daec`, una política de exportación para el qtree llamado `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`, y una política de exportación vacía llamada `trident_empty` en la SVM. Las reglas para la política de exportación del FlexVol serán un superconjunto de cualquier regla contenida en las políticas de exportación del qtree. La política de exportación vacía se reutilizará para cualquier volumen que no esté adjunto.

- `autoExportCIDRs` contiene una lista de bloques de direcciones. Este campo es opcional y su valor predeterminado es ["0.0.0.0/0", "::/0"]. Si no se define, Trident agrega todas las direcciones unicast de alcance global que se encuentran en los nodos de trabajo con publicaciones.

En este ejemplo, el espacio de direcciones `192.168.0.0/24` se proporciona. Esto indica que las direcciones IP de los nodos de Kubernetes que estén dentro de este rango de direcciones con publicaciones se añadirán a la política de exportación que crea Trident. Cuando Trident registra un nodo en el que se ejecuta, recupera las direcciones IP del nodo y las compara con los bloques de direcciones proporcionados en `autoExportCIDRs`. Al momento de publicar, después de filtrar las IP, Trident crea las reglas de la política de exportación para las direcciones IP de cliente del nodo al que está publicando.

Puedes actualizar `autoExportPolicy` y `autoExportCIDRs` para los backends después de crearlos. Puedes añadir nuevos CIDR para un backend que se gestiona automáticamente o eliminar los CIDR existentes. Ten cuidado al eliminar los CIDR para asegurarte de que no se interrumpan las conexiones existentes. También puedes elegir deshabilitar `autoExportPolicy` para un backend y volver a una política de exportación creada manualmente. Esto requerirá configurar el parámetro `exportPolicy` en la configuración del backend.

Después de que Trident crea o actualiza un backend, puedes verificar el backend usando `tridentctl` o el correspondiente `tridentbackend` CRD:

```
./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileSystemType: ext4
```

Al eliminar un nodo, Trident revisa todas las políticas de exportación para eliminar las reglas de acceso correspondientes al nodo. Al eliminar esta dirección IP de nodo de las políticas de exportación de los backends administrados, Trident evita montajes no autorizados, a menos que esta IP sea reutilizada por un nuevo nodo en el clúster.

Para los backends preexistentes, actualizar el backend con `tridentctl update backend` asegura que Trident gestione las políticas de exportación automáticamente. Esto crea dos nuevas políticas de exportación nombradas según el UUID y el nombre del qtree del backend cuando sean necesarias. Los volúmenes que están presentes en el backend usarán las nuevas políticas de exportación después de desmontarlos y volver a montarlos.

| | |
|---|---|
| **NOTA** | Al eliminar un backend con políticas de exportación autogestionadas, se eliminará la política de exportación creada dinámicamente. Si el backend se vuelve a crear, se trata como un backend nuevo y resultará en la creación de una nueva política de exportación. |

Si se actualiza la dirección IP de un nodo activo, debes reiniciar el pod de Trident en el nodo. Trident actualizará la política de exportación para los backends que administra para reflejar este cambio de IP.

#### Prepárate para aprovisionar volúmenes SMB

Con un poco de preparación adicional, puedes aprovisionar volúmenes SMB usando `ontap-nas`drivers.

| | |
|---|---|
| **ADVERTENCIA** | Debes configurar tanto los protocolos NFS como SMB/CIFS en la SVM para crear un `ontap-nas-economy` volumen SMB para clústeres ONTAP locales. Si no configuras alguno de estos protocolos, la creación del volumen SMB fallará. |

**NOTA** `autoExportPolicy` no es compatible con volúmenes SMB.

**Antes de empezar**

Antes de que puedas aprovisionar volúmenes SMB, debes tener lo siguiente.

- Un clúster de Kubernetes con un nodo controlador Linux y al menos un nodo trabajador Windows que ejecuta Windows Server 2022. Trident solo admite volúmenes SMB montados en pods que se ejecutan en nodos Windows.
- Al menos un secreto de Trident con tus credenciales de Active Directory. Para generar un secreto `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Un proxy CSI configurado como un servicio de Windows. Para configurar un `csi-proxy`, consulta "GitHub: CSI Proxy" o "GitHub: CSI Proxy para Windows" para nodos de Kubernetes que se ejecutan en Windows.

**Pasos**

1. Para ONTAP local, puedes crear opcionalmente un recurso compartido SMB o Trident puede crear uno para ti.

   **NOTA** Se requieren recursos compartidos SMB para Amazon FSx for ONTAP.

   Puedes crear los recursos compartidos de administrador de SMB de dos maneras: usando el complemento "Microsoft Management Console" Shared Folders o usando la CLI de ONTAP. Para crear los recursos compartidos de SMB usando la CLI de ONTAP:

   a. Si es necesario, crea la estructura de ruta del directorio para el recurso compartido.

      El `vserver cifs share create`comando comprueba la ruta especificada en la opción -path durante la creación del recurso compartido. Si la ruta especificada no existe, el comando falla.

   b. Crea un recurso compartido SMB asociado con la SVM especificada:

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

   c. Verifica que se creó el recurso compartido:

```
vserver cifs share show -share-name share_name
```

   **NOTA** Consulta "Crear un recurso compartido SMB" para obtener detalles completos.

2. Al crear el backend, debes configurar lo siguiente para especificar los volúmenes SMB. Para todas las

opciones de configuración del backend de FSx for ONTAP, consulta "Opciones de configuración de FSx for ONTAP y ejemplos".

| Parámetro | Descripción | Ejemplo |
|---|---|---|
| `smbShare` | Puedes especificar uno de los siguientes: el nombre de un recurso compartido SMB creado usando Microsoft Management Console o la CLI de ONTAP, un nombre para permitir que Trident cree el recurso compartido SMB, o puedes dejar el parámetro en blanco para evitar el acceso compartido común a los volúmenes. Este parámetro es opcional para ONTAP local. Este parámetro es obligatorio para los backends de Amazon FSx for ONTAP y no puede estar en blanco. | `smb-share` |
| `nasType` | **Debe establecerse en `smb`.** Si es nulo, el valor predeterminado es `nfs`. | `smb` |
| `securityStyle` | Estilo de seguridad para nuevos volúmenes. **Debe configurarse en `ntfs` o `mixed` para volúmenes SMB.** | `ntfs` o `mixed` para volúmenes SMB |
| `unixPermissions` | Modo para nuevos volúmenes. **Debe dejarse vacío para volúmenes SMB.** | "" |

### Habilita SMB seguro

A partir de la versión 25.06, NetApp Trident admite el aprovisionamiento seguro de volúmenes SMB creados mediante `ontap-nas` y `ontap-nas-economy` backends. Cuando SMB seguro está habilitado, puedes proporcionar acceso controlado a los recursos compartidos SMB para usuarios y grupos de usuarios de Active Directory (AD) usando listas de control de acceso (ACL).

#### Puntos para recordar

- No se admite la importación `ontap-nas-economy` de volúmenes.
- Solo se admiten clones de solo lectura para `ontap-nas-economy` volúmenes.
- Si Secure SMB está habilitado, Trident ignorará el recurso compartido SMB mencionado en el backend.
- Actualizar la anotación de PVC, la anotación de la storage class y el campo backend no actualiza la ACL del recurso compartido SMB.
- La ACL de recurso compartido SMB especificada en la anotación del PVC clonado tendrá prioridad sobre las del PVC de origen.
- Asegúrate de proporcionar usuarios de AD válidos al habilitar SMB seguro. Los usuarios no válidos no se añadirán a la ACL.
- Si proporcionas el mismo usuario de AD en el backend, la clase de almacenamiento y el PVC con diferentes permisos, la prioridad de permisos será: PVC, clase de almacenamiento y luego backend.
- SMB seguro es compatible para `ontap-nas`importaciones de volúmenes administrados y no aplica a importaciones de volúmenes no administrados.

#### Pasos

1. Especifica adAdminUser en TridentBackendConfig como se muestra en el siguiente ejemplo:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. Añade la anotación en la clase de almacenamiento.

   Agrega la `trident.netapp.io/smbShareAdUser` anotación a la clase de almacenamiento para habilitar SMB seguro sin fallar. El valor de usuario especificado para la anotación `trident.netapp.io/smbShareAdUser` debe ser el mismo que el nombre de usuario especificado en el `smbcreds` secreto. Puedes elegir una de las siguientes para `smbShareAdUserPermission`: `full_control`, `change` o `read`. El permiso predeterminado es `full_control`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1. Crea un PVC.

   El siguiente ejemplo crea una PVC:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

#### Opciones de configuración y ejemplos de ONTAP NAS

Aprende a crear y usar controladores NAS de ONTAP con tu instalación de Trident. Esta sección ofrece ejemplos de configuración de backends y detalles para asignar backends a StorageClasses. A partir de la versión 25.10, NetApp Trident es compatible con "NetApp sistemas de almacenamiento AFX". Los sistemas de almacenamiento NetApp AFX se diferencian de otros sistemas basados en ONTAP (ASA, AFF y FAS) en la implementación de su capa de almacenamiento.

| | |
|---|---|
| **NOTA** | Sólo el `ontap-nas` controlador (con protocolo NFS) es compatible con NetApp AFX systems; el protocolo SMB no es compatible. |

##### Opciones de configuración del backend

En la configuración del backend de Trident, no necesitas especificar que tu sistema es un sistema de almacenamiento AFX de NetApp. Cuando seleccionas `ontap-nas` como `storageDriverName`, Trident detecta automáticamente el sistema de almacenamiento AFX. Algunos parámetros de configuración del backend no aplican a los sistemas de almacenamiento AFX.

La siguiente tabla muestra las opciones de configuración del backend:

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `version` | | Siempre 1 |

| Parámetro | Descripción | Predeterminado |
| --- | --- | --- |
| `storageDriverName` | Nombre del controlador de almacenamiento<br>**NOTA** Para los sistemas AFX de NetApp, solo `ontap-nas` es compatible. | `ontap-nas`, `ontap-nas-economy`, o `ontap-nas-flexgroup` |
| `backendName` | Nombre personalizado o el backend de almacenamiento | Nombre del driver + "_" + dataLIF |
| `managementLIF` | Dirección IP de un clúster o de una LIF de administración de SVM. Se puede especificar un nombre de dominio completo (FQDN). Se puede configurar para usar direcciones IPv6 si Trident se instaló usando el flag de IPv6. Las direcciones IPv6 deben definirse entre corchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Para una conmutación de sitios sin interrupciones de MetroCluster, consulta el Ejemplo de MetroCluster. | "10.0.0.1", "[2001:1234:abcd::fefe]" |
| `dataLIF` | Dirección IP del LIF de protocolo. NetApp recomienda especificar `dataLIF`. Si no se proporciona, Trident obtiene los dataLIF del SVM. Puedes especificar un nombre de dominio completo (FQDN) para las operaciones de montaje NFS, lo que te permite crear un DNS de round-robin para balancear la carga entre varios dataLIF. Se puede cambiar después de la configuración inicial. Consulta . Se puede configurar para usar direcciones IPv6 si Trident se instaló usando el flag de IPv6. Las direcciones IPv6 deben definirse entre corchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **Omite para MetroCluster.** Consulta la Ejemplo de MetroCluster. | Dirección especificada o derivada de SVM, si no se especifica (no recomendado) |
| `svm` | Máquina virtual de almacenamiento a usar **Omitir para MetroCluster.** Consulta la Ejemplo de MetroCluster. | Se deriva si se especifica un SVM `managementLIF` |
| `autoExportPolicy` | Habilita la creación y actualización automática de políticas de exportación [Boolean]. Usando las opciones `autoExportPolicy` y `autoExportCIDRs`, Trident puede gestionar las políticas de exportación automáticamente. | false |
| `autoExportCIDRs` | Lista de CIDR para filtrar las direcciones IP de los nodos de Kubernetes cuando `autoExportPolicy` está habilitado. Usando las opciones `autoExportPolicy` y `autoExportCIDRs`, Trident puede gestionar las políticas de exportación automáticamente. | ["0.0.0.0/0", "::/0"]` |
| `labels` | Conjunto de etiquetas arbitrarias con formato JSON para aplicar en volúmenes | "" |

| Parámetro | Descripción | Predeterminado |
| --- | --- | --- |
| `clientCertificate` | Valor codificado en Base64 del certificado del cliente. Usado para auth basada en certificados | "" |
| `clientPrivateKey` | Valor codificado en Base64 de la clave privada del cliente. Usado para auth basada en certificados | "" |
| `trustedCACertificate` | Valor codificado en Base64 del certificado de CA de confianza. Opcional. Usado para auth basada en certificados | "" |
| `username` | Nombre de usuario para conectarse al clúster/SVM. Se usa para la autenticación basada en credenciales. Para la autenticación de Active Directory, consulta "Autentica Trident en un SVM backend usando credenciales de Active Directory". | |
| `password` | Contraseña para conectarte al clúster/SVM. Se usa para la autenticación basada en credenciales. Para la autenticación de Active Directory, consulta "Autentica Trident en un SVM backend usando credenciales de Active Directory". | |
| `storagePrefix` | Prefijo utilizado al aprovisionar nuevos volúmenes en la SVM. No se puede actualizar después de configurarlo<br>**NOTA** Al usar ontap-nas-economy y un storagePrefix que tiene 24 caracteres o más, los qtrees no tendrán el prefijo de almacenamiento incorporado, aunque estará en el nombre del volumen. | "Trident" |

| Parámetro | Descripción | Predeterminado |
| --- | --- | --- |
| `aggregate` | Agregado para aprovisionamiento (opcional; si se configura, debe asignarse a la SVM). Para el `ontap-nas-flexgroup` driver, esta opción se ignora. Si no se asigna, cualquiera de los agregados disponibles se puede usar para aprovisionar un FlexGroup volumen.<br>**NOTA**<br>Cuando el agregado se actualiza en SVM, se actualiza automáticamente en Trident mediante el sondeo de SVM sin tener que reiniciar el Trident Controller. Cuando has configurado un agregado específico en Trident para aprovisionar volúmenes, si el agregado se renombra o se mueve fuera de la SVM, el backend pasará a estado fallido en Trident mientras sondea el agregado de la SVM. Debes cambiar el agregado por uno que esté presente en la SVM o eliminarlo por completo para que el backend vuelva a estar en línea.<br>**No especifiques para sistemas de almacenamiento AFX.** | "" |
| `limitAggregateUsage` | Falla el aprovisionamiento si el uso es superior a este porcentaje. **No aplica a Amazon FSx for ONTAP**. **No especifiques para sistemas de almacenamiento AFX.** | "" (no aplicado por defecto) |

| Parámetro | Descripción | Predeterminado |
| --- | --- | --- |
| flexgroupAggregateList | Lista de agregados para aprovisionamiento (opcional; si se configura, debe asignarse a la SVM). Todos los agregados asignados a la SVM se utilizan para aprovisionar un volumen FlexGroup. Compatible con el controlador de almacenamiento **ontap-nas-flexgroup**.<br>**NOTA** Cuando la lista de agregados se actualiza en SVM, la lista se actualiza en Trident automáticamente mediante el sondeo de SVM sin tener que reiniciar el controlador Trident. Cuando has configurado una lista de agregados específica en Trident para aprovisionar volúmenes, si la lista de agregados se renombra o se mueve fuera de SVM, el backend pasará a estado de error en Trident mientras sondea el agregado de SVM. Debes cambiar la lista de agregados por una que esté presente en SVM o eliminarla por completo para que el backend vuelva a estar en línea. | "" |
| `limitVolumeSize` | Falla el aprovisionamiento si el tamaño del volumen solicitado supera este valor. | "" (no aplicado por defecto) |
| `debugTraceFlags` | Indicadores de depuración para utilizar cuando estés solucionando problemas. Por ejemplo, {"api":false, "method":true} no uses `debugTraceFlags` a menos que estés resolviendo problemas y necesites un volcado de registro detallado. | null |
| `nasType` | Configura la creación de volúmenes NFS o SMB. Las opciones son `nfs`, `smb` o null. Si lo configuras en null, se usarán volúmenes NFS por defecto. **Si se especifica, siempre se establece en `nfs` para sistemas de almacenamiento AFX**. | `nfs` |
| `nfsMountOptions` | Lista de opciones de montaje NFS separadas por comas. Las opciones de montaje para volúmenes persistentes de Kubernetes normalmente se especifican en las clases de almacenamiento, pero si no se especifican opciones de montaje en una clase de almacenamiento, Trident usará las opciones de montaje especificadas en el archivo de configuración del backend de almacenamiento. Si no se especifican opciones de montaje en la clase de almacenamiento ni en el archivo de configuración, Trident no establecerá ninguna opción de montaje en un volumen persistente asociado. | "" |

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `qtreesPerFlexvol` | Máximo de Qtrees por FlexVol, debe estar en el rango [50, 300] | "200" |
| `smbShare` | Puedes especificar uno de los siguientes: el nombre de un recurso compartido SMB creado usando Microsoft Management Console o la CLI de ONTAP, un nombre para permitir que Trident cree el recurso compartido SMB, o puedes dejar el parámetro en blanco para evitar el acceso compartido común a los volúmenes. Este parámetro es opcional para ONTAP local. Este parámetro es obligatorio para los backends de Amazon FSx for ONTAP y no puede estar en blanco. | `smb-share` |
| `useREST` | Parámetro booleano para usar las ONTAP REST APIs. `useREST`Cuando se establece en `true`, Trident usa las ONTAP REST APIs para comunicarse con el backend; cuando se establece en `false`, Trident usa llamadas ONTAPI (ZAPI) para comunicarse con el backend. Esta función requiere ONTAP 9.11.1 y versiones posteriores. Además, el rol de inicio de sesión de ONTAP utilizado debe tener acceso a la aplicación `ontapi`. Esto se cumple con los roles predefinidos `vsadmin` y `cluster-admin`. A partir de la versión Trident 24.06 y ONTAP 9.15.1 o posteriores, `useREST`está configurado en `true`de forma predeterminada; cambia `useREST` a `false` para usar llamadas ONTAPI (ZAPI). **Si se especifica, siempre se establece en `true` para sistemas de almacenamiento AFX**. | `true` para ONTAP 9.15.1 o posterior, de lo contrario `false`. |
| `limitVolumePoolSize` | Tamaño máximo solicitable de FlexVol al utilizar Qtrees en el backend ontap-nas-economy. | "" (no aplicado por defecto) |
| `denyNewVolumePools` | Restringe `ontap-nas-economy` a los backends crear nuevos volúmenes FlexVol para contener sus Qtrees. Solo se usan FlexVols preexistentes para aprovisionar nuevos PV. | |
| `adAdminUser` | Usuario o grupo de usuarios administradores de Active Directory con acceso total a los recursos compartidos SMB. Usa este parámetro para otorgar derechos de administrador al recurso compartido SMB con control total. | |

#### Opciones de configuración de backend para aprovisionar volúmenes

Puedes controlar el aprovisionamiento predeterminado usando estas opciones en la `defaults` sección de la configuración. Por ejemplo, mira los ejemplos de configuración abajo.

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `spaceAllocation` | Asignación de espacio para Qtrees | "true" |

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `spaceReserve` | Modo de reserva de espacio; "ninguno" (fino) o "volumen" (grueso) | "none" |
| `snapshotPolicy` | Política de SnapVault a utilizar | "none" |
| `qosPolicy` | Grupo de políticas de QoS para asignar a los volúmenes creados. Elige uno de qosPolicy o adaptiveQosPolicy por cada pool de almacenamiento/backend | "" |
| `adaptiveQosPolicy` | Grupo de políticas de QoS adaptativo para asignar a los volúmenes creados. Elige uno de qosPolicy o adaptiveQosPolicy por pool de almacenamiento/backend. No compatible con ontap-nas-economy. | "" |
| `snapshotReserve` | Porcentaje de volumen reservado para instantáneas | "0" si `snapshotPolicy` es "none", de lo contrario "" |
| `splitOnClone` | Divide un clon de su padre al momento de su creación | "false" |
| `encryption` | Habilita NetApp Volume Encryption (NVE) en el nuevo volumen; el valor predeterminado es `false`. NVE debe tener licencia y estar habilitado en el clúster para usar esta opción. Si NAE está habilitado en el backend, cualquier volumen aprovisionado en Trident tendrá NAE habilitado. Para más información, consulta: "Cómo funciona Trident con NVE y NAE". | "false" |
| `tieringPolicy` | Política de organización en niveles para usar "none" | |
| `unixPermissions` | Modo para nuevos volúmenes | "777" para volúmenes NFS; vacío (no aplicable) para volúmenes SMB |
| `snapshotDir` | Controla el acceso al `.snapshot` directorio | `true`, `false` (establecer explícitamente). |
| `exportPolicy` | Política de exportación a usar | "default" |
| `securityStyle` | Estilo de seguridad para nuevos volúmenes. NFS admite `mixed` y `unix` estilos de seguridad. SMB admite `mixed` y `ntfs` estilos de seguridad. | NFS predeterminado es `unix`. SMB predeterminado es `ntfs`. |
| `nameTemplate` | Plantilla para crear nombres de volúmenes personalizados. | "" |

**NOTA** Usar grupos de políticas de QoS con Trident requiere ONTAP 9.8 o una versión posterior. Deberías usar un grupo de políticas de QoS no compartido y asegurarte de que el grupo de políticas se aplique a cada componente individualmente. Un grupo de políticas de QoS compartido impone el límite máximo para el rendimiento total de todas las cargas de trabajo.

#### Ejemplos de aprovisionamiento de volumen

Aquí tienes un ejemplo con valores predeterminados definidos:

```
---
version: 1
storageDriverName: ontap-nas
backendName: customBackendName
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: dev1
  backend: dev1-nasbackend
svm: trident_svm
username: cluster-admin
password: <password>
limitAggregateUsage: 80%
limitVolumeSize: 50Gi
nfsMountOptions: nfsvers=4
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: premium
  exportPolicy: myk8scluster
  snapshotPolicy: default
  snapshotReserve: "10"
```

Para `ontap-nas` y `ontap-nas-flexgroups`, Trident ahora usa un nuevo cálculo para asegurarse de que el FlexVol tenga el tamaño correcto con el porcentaje de snapshotReserve y el PVC. Cuando el usuario solicita un PVC, Trident crea el FlexVol original con más espacio usando el nuevo cálculo. Este cálculo asegura que el usuario reciba el espacio escribible que pidió en el PVC, y no menos espacio del que solicitó. Antes de la v21.07, cuando el usuario solicitaba un PVC (por ejemplo, 5 GiB), con el snapshotReserve al 50 por ciento, solo obtenía 2.5 GiB de espacio escribible. Esto es porque lo que el usuario pedía era todo el volumen y `snapshotReserve` es un porcentaje de eso. Con Trident 21.07, lo que el usuario pide es el espacio escribible y Trident define el número de `snapshotReserve` como el porcentaje del volumen completo. Esto no aplica a `ontap-nas-economy`. Mira el siguiente ejemplo para ver cómo funciona esto:

El cálculo es el siguiente:

```
Total volume size = <PVC requested size> / (1 - (<snapshotReserve
percentage> / 100))
```

Para snapshotReserve = 50%, y una solicitud de PVC de 5 GiB, el tamaño total del volumen es 5/0.5 = 10 GiB y el tamaño disponible es 5 GiB, que es lo que el usuario pidió en la solicitud de PVC. El `volume show` comando debería mostrar resultados similares a este ejemplo:

```
Vserver   Volume       Aggregate    State      Type       Size  Available Used%
--------- ------------ ------------ ---------- ---- ---------- ---------- -----
                  _pvc_89f1c156_3801_4de4_9f9d_034d54c395f4
                                   online     RW         10GB     5.00GB    0%
                  _pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba
                                   online     RW          1GB     511.8MB    0%
2 entries were displayed.
```

Los backends existentes de instalaciones anteriores aprovisionarán volúmenes como se explicó arriba cuando actualices Trident. Para los volúmenes que creaste antes de actualizar, deberías redimensionar sus volúmenes para que se note el cambio. Por ejemplo, un PVC de 2 GiB con `snapshotReserve=50` antes generaba un volumen que proporcionaba 1 GiB de espacio escribible. Redimensionar el volumen a 3 GiB, por ejemplo, le da a la aplicación 3 GiB de espacio escribible en un volumen de 6 GiB.

#### Ejemplos de configuración mínima

Los siguientes ejemplos muestran configuraciones básicas que dejan la mayoría de los parámetros en sus valores predeterminados. Esta es la forma más fácil de definir un backend.

| NOTA | Si estás usando Amazon FSx en NetApp ONTAP con Trident, la recomendación es especificar nombres DNS para los LIF en vez de direcciones IP. |
|---|---|

**Ejemplo de ONTAP NAS economy**

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

**Ejemplo de ONTAP NAS FlexGroup**

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

#### Ejemplo de MetroCluster

Puedes configurar el backend para evitar tener que actualizar manualmente la definición del backend después de la conmutación de sitios y la conmutación de vuelta durante "Replicación y recuperación de SVM".

Para una conmutación de sitios sin interrupciones, especifica la SVM usando `managementLIF` y omite los parámetros `dataLIF` y `svm`. Por ejemplo:

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

#### Ejemplo de volúmenes SMB

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
nasType: smb
securityStyle: ntfs
unixPermissions: ""
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

#### Ejemplo de autenticación basada en certificados

Este es un ejemplo de configuración mínima de backend. `clientCertificate`, `clientPrivateKey` y `trustedCACertificate` (opcional, si usas una CA de confianza) se rellenan en `backend.json` y toman los valores codificados en base64 del certificado de cliente, la clave privada y el certificado de CA de confianza, respectivamente.

```
---
version: 1
backendName: DefaultNASBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

#### Ejemplo de política de exportación automática

Este ejemplo muestra cómo puedes indicarle a Trident que use políticas de exportación dinámicas para crear y gestionar la política de exportación automáticamente. Esto funciona igual para los controladores `ontap-nas-economy` y `ontap-nas-flexgroup`.

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-nasbackend
autoExportPolicy: true
autoExportCIDRs:
- 10.0.0.0/24
username: admin
password: password
nfsMountOptions: nfsvers=4
```

#### Ejemplo de direcciones IPv6

Este ejemplo muestra `managementLIF` usando una dirección IPv6.

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas_ipv6_backend
managementLIF: "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]"
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-ontap-ipv6
svm: nas_ipv6_svm
username: vsadmin
password: password
```

#### Ejemplo de Amazon FSx for ONTAP usando volúmenes SMB

El `smbShare` parámetro es necesario para FSx for ONTAP usando volúmenes SMB.

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

#### Ejemplo de configuración de backend con nameTemplate

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap-nas-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

#### Ejemplos de backends con pools virtuales

En los archivos de definición de backend de ejemplo que se muestran a continuación, se establecen valores predeterminados específicos para todos los pools de almacenamiento, como `spaceReserve` en ninguno, `spaceAllocation` en falso y `encryption` en falso. Los grupos virtuales se definen en la sección de almacenamiento.

Trident establece las etiquetas de aprovisionamiento en el campo "Comentarios". Los comentarios se establecen en FlexVol para `ontap-nas` o en FlexGroup para `ontap-nas-flexgroup`. Trident copia todas las etiquetas presentes en un pool virtual al volumen de almacenamiento al aprovisionar. Para mayor comodidad, los administradores de almacenamiento pueden definir etiquetas por pool virtual y agrupar volúmenes por etiqueta.

En estos ejemplos, algunos de los pools de almacenamiento establecen sus propios `spaceReserve`, `spaceAllocation` y `encryption` valores, y algunos pools anulan los valores predeterminados.

**Ejemplo de ONTAP NAS**

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
nfsMountOptions: nfsvers=4
defaults:
  spaceReserve: none
  encryption: "false"
  qosPolicy: standard
labels:
  store: nas_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      app: msoffice
      cost: "100"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
      adaptiveQosPolicy: adaptive-premium
  - labels:
      app: slack
      cost: "75"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: legal
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      app: wordpress
```

```
      cost: "50"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      app: mysqldb
      cost: "25"
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

**Ejemplo de ONTAP NAS FlexGroup**

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: flexgroup_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "50000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: gold
      creditpoints: "30000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      protection: bronze
      creditpoints: "10000"
    zone: us_east_1d
    defaults:
```

```
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

**Ejemplo de ONTAP NAS economy**

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: nas_economy_store
region: us_east_1
storage:
  - labels:
      department: finance
      creditpoints: "6000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: engineering
      creditpoints: "3000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      department: humanresource
      creditpoints: "2000"
    zone: us_east_1d
    defaults:
      spaceReserve: volume
```

```
      encryption: "false"
      unixPermissions: "0775"
```

#### Asigna backends a StorageClasses

Las siguientes definiciones de StorageClass hacen referencia a Ejemplos de backends con pools virtuales. Usando el campo `parameters.selector`, cada StorageClass indica qué grupos virtuales pueden usarse para alojar un volumen. El volumen tendrá los aspectos definidos en el grupo virtual elegido.

- El `protection-gold` StorageClass se asignará al primer y segundo pool virtual en el `ontap-nas-flexgroup` backend. Estos son los únicos pools que ofrecen protección de nivel oro.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- El `protection-not-gold` StorageClass se asignará al tercer y cuarto pool virtual en el `ontap-nas-flexgroup` backend. Estos son los únicos pools que ofrecen un nivel de protección distinto al gold.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- El `app-mysqldb` StorageClass se mapeará al cuarto pool virtual en el `ontap-nas` backend. Este es el único pool que ofrece configuración de pool de almacenamiento para app tipo mysqldb.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- El `protection-silver-creditpoints-20k` StorageClass se asignará al tercer pool virtual en el `ontap-nas-flexgroup` backend. Este es el único pool que ofrece protección de nivel plata y 20000 puntos de crédito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- El `creditpoints-5k` StorageClass se asignará al tercer pool virtual en el `ontap-nas` backend y al segundo pool virtual en el `ontap-nas-economy` backend. Estas son las únicas ofertas de grupos con 5000 puntos de crédito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Trident decidirá qué grupo virtual se selecciona y se asegurará de que se cumpla el requisito de almacenamiento.

#### Actualiza `dataLIF` después de la configuración inicial

Puedes cambiar el dataLIF después de la configuración inicial ejecutando el siguiente comando para proporcionar el nuevo archivo JSON del backend con el dataLIF actualizado.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

**NOTA** Si los PVC están conectados a uno o varios pods, tienes que desactivar todos los pods correspondientes y luego volverlos a activar para que el nuevo dataLIF surta efecto.

#### Ejemplos seguros de SMB

##### Configuración del backend con el controlador ontap-nas

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

##### Configuración del backend con el controlador ontap-nas-economy

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

**Configuración de backend con storage pool**

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret
```

**Ejemplo de clase de almacenamiento con el controlador ontap-nas**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

**NOTA** Asegúrate de agregar `annotations` para habilitar SMB seguro. SMB seguro no funciona sin las anotaciones, independientemente de las configuraciones establecidas en el Backend o PVC.

**Ejemplo de clase de almacenamiento con el controlador ontap-nas-economy**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

**Ejemplo de PVC con un único usuario AD**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

**Ejemplo de PVC con varios usuarios AD**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

## Amazon FSx for NetApp ONTAP

#### Usa Trident con Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP" es un servicio de AWS totalmente administrado que ejecuta sistemas de archivos impulsados por el sistema operativo de almacenamiento NetApp ONTAP. Proporciona características, rendimiento y administración de ONTAP con la escalabilidad y la simplicidad operativa de AWS. Un sistema de archivos es el recurso principal en Amazon FSx y es análogo a un clúster ONTAP local. Cada sistema de archivos contiene una o varias máquinas virtuales de almacenamiento (SVM), y cada SVM contiene uno o varios volúmenes que almacenan archivos y directorios. Esta integración permite a los clústeres de Kubernetes que se ejecutan en Amazon Elastic Kubernetes Service (EKS) aprovisionar volúmenes persistentes respaldados por ONTAP para cargas de trabajo de bloques y archivos.

#### Requisitos

Además de "Requisitos de Trident", para integrar FSx for ONTAP con Trident, necesitas:

- Un clúster de Amazon EKS existente o un clúster de Kubernetes autoadministrado con `kubectl` instalado.
- Un sistema de archivos Amazon FSx para NetApp ONTAP existente y una máquina virtual de almacenamiento (SVM) a la que se puede acceder desde los nodos de trabajo de tu clúster.
- Nodos worker que están preparados para "NFS o iSCSI".

> **NOTA** Asegúrate de seguir los pasos de preparación de nodo necesarios para Amazon Linux y Ubuntu "Amazon Machine Images" (AMIs) según tu tipo de AMI de EKS.

#### Consideraciones

- Volúmenes SMB:
  - Los volúmenes SMB son compatibles usando solo el `ontap-nas` driver.
  - Los volúmenes SMB no son compatibles con el complemento Trident EKS.
  - Trident solo admite volúmenes SMB montados en pods que se ejecutan en nodos Windows. Consulta "Prepárate para aprovisionar volúmenes SMB" para más detalles.
- Antes de Trident 24.02, los volúmenes creados en sistemas de archivos de Amazon FSx que tenían copias de seguridad automáticas habilitadas no podían ser eliminados por Trident. Para evitar este problema en Trident 24.02 o versiones posteriores, especifica el `fsxFilesystemID`, AWS `apiRegion`, AWS `apikey` y AWS `secretKey` en el archivo de configuración del backend para AWS FSx for ONTAP.

> **NOTA** Si estás especificando una función de IAM para Trident, entonces puedes omitir especificar los campos `apiRegion`, `apiKey` y `secretKey` explícitamente para Trident. Para obtener más información, consulta "Opciones de configuración de FSx for ONTAP y ejemplos".

### Uso simultáneo de Trident SAN/iSCSI y el controlador EBS-CSI

Si planeas usar controladores ontap-san (por ejemplo, iSCSI) con AWS (EKS, ROSA, EC2 o cualquier otra instancia), la configuración de multivía requerida en los nodos podría entrar en conflicto con el controlador CSI de Amazon Elastic Block Store (EBS). Para asegurarte de que la multivía funcione sin interferir con los discos EBS en el mismo nodo, necesitas excluir EBS en tu configuración de multivía. Este ejemplo muestra un `multipath.conf` archivo que incluye la configuración requerida de Trident y excluye los discos EBS de la multivía:

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

#### Autenticación

Trident ofrece dos modos de autenticación.

- Basado en credenciales (recomendado): almacena las credenciales de forma segura en AWS Secrets Manager. Puedes usar el `fsxadmin` usuario para tu sistema de archivos o el `vsadmin` usuario configurado para tu SVM.

| | |
|---|---|
| **ADVERTENCIA** | Trident espera ejecutarse como un `vsadmin` usuario de SVM o como un usuario con un nombre diferente que tenga el mismo rol. Amazon FSx for NetApp ONTAP tiene un `fsxadmin` usuario que es un reemplazo limitado del usuario de clúster de ONTAP `admin`. Te recomendamos mucho usar `vsadmin` con Trident. |

- Basado en certificado: Trident se comunicará con la SVM en tu sistema de archivos FSx usando un certificado instalado en tu SVM.

Para obtener detalles sobre cómo habilitar la autenticación, consulta la autenticación para tu tipo de controlador:

- "Autenticación NAS de ONTAP"
- "Autenticación SAN de ONTAP"

#### Imágenes de máquina de Amazon (AMIs) probadas

El clúster EKS es compatible con varios sistemas operativos, pero AWS ha optimizado ciertas Amazon Machine Images (AMIs) para contenedores y EKS. Las siguientes AMIs se han probado con NetApp Trident 25.02.

| AMI | NAS | NAS-economy | iSCSI | iSCSI-economy |
|---|---|---|---|---|
| AL2023_x86_64_STANDARD | Sí | Sí | Sí | Sí |
| AL2_x86_64 | Sí | Sí | Sí* | Sí* |
| BOTTLEROCKET_x86_64 | Sí** | Sí | N/A | N/A |
| AL2023_ARM_64_STANDARD | Sí | Sí | Sí | Sí |

| | | | | |
|---|---|---|---|---|
| AL2_ARM_64 | Sí | Sí | Sí* | Sí* |
| BOTTLEROCKET_ARM_64 | Sí** | Sí | N/A | N/A |

- * No se puede eliminar el PV sin reiniciar el nodo
- ** No funciona con NFSv3 con Trident versión 25.02.

| | |
|---|---|
| **NOTA** | Si la AMI que buscas no aparece aquí, no significa que no sea compatible; simplemente significa que no se ha probado. Esta lista sirve como guía para las AMI que se sabe que funcionan. |

**Pruebas realizadas con**:

- Versión de EKS: 1.32
- Método de instalación: Helm 25.06 y como AWS add-On 25.06
- Para NAS se probaron tanto NFSv3 como NFSv4.1.
- Para SAN solo se probó iSCSI, no NVMe-oF.

**Pruebas realizadas**:

- Crear: clase de almacenamiento, pvc, pod
- Eliminar: pod, pvc (regular, qtree/lun – economy, NAS con backup de AWS)

#### Encuentra más información

- "Documentación de Amazon FSx para NetApp ONTAP"
- "Entrada de blog sobre Amazon FSx para NetApp ONTAP"

### Crea un rol de IAM y un secreto de AWS

Puedes configurar pods de Kubernetes para acceder a recursos de AWS autenticándote como un rol de AWS IAM en lugar de proporcionar credenciales de AWS explícitas.

| | |
|---|---|
| **NOTA** | Para autenticarse mediante una función de AWS IAM, debes tener un clúster de Kubernetes implementado usando EKS. |

#### Crear secreto de AWS Secrets Manager

Como Trident va a emitir APIs contra un vserver de FSx para gestionar el almacenamiento por ti, necesitará credenciales para hacerlo. La forma segura de pasar esas credenciales es a través de un secreto de AWS Secrets Manager. Así que, si aún no tienes uno, tendrás que crear un secreto de AWS Secrets Manager que contenga las credenciales para la cuenta vsadmin.

Este ejemplo crea un secreto de AWS Secrets Manager para almacenar las credenciales de Trident CSI:

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

#### Crear política de IAM

Trident también necesita permisos de AWS para funcionar correctamente. Por lo tanto, necesitas crear una política que le dé a Trident los permisos que necesita.

Los siguientes ejemplos crean una política de IAM mediante el AWS CLI:

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy
-document file://policy.json
  --description "This policy grants access to Trident CSI to FSxN and
Secrets manager"
```

**Ejemplo de política JSON**:

```
{
  "Statement": [
    {
      "Action": [
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes",
        "fsx:CreateVolume",
        "fsx:RestoreVolumeFromSnapshot",
        "fsx:DescribeStorageVirtualMachines",
        "fsx:UntagResource",
        "fsx:UpdateVolume",
        "fsx:TagResource",
        "fsx:DeleteVolume"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "secretsmanager:GetSecretValue",
      "Effect": "Allow",
      "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-
id>:secret:<aws-secret-manager-name>*"
    }
  ],
  "Version": "2012-10-17"
}
```

**Crea una identidad de pod o un rol de IAM para la asociación de cuentas de servicio (IRSA)**

Puedes configurar una cuenta de servicio de Kubernetes para que asuma un rol de AWS Identity and Access Management (IAM) con EKS Pod Identity o IAM role for Service account association (IRSA). Cualquier pod que esté configurado para usar la cuenta de servicio podrá acceder a cualquier servicio de AWS al que el rol tenga permisos de acceso.

**Identidad de pod**

Las asociaciones de Amazon EKS Pod Identity proporcionan la capacidad de administrar credenciales para tus aplicaciones, de forma similar a como los perfiles de instancias de Amazon EC2 proporcionan credenciales a las instancias de Amazon EC2.

**Instala Pod Identity en tu clúster EKS**:

Puedes crear una identidad de Pod a través de la consola de AWS o usando el siguiente comando de la AWS CLI:

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

Para más información consulta "Configura el agente de identidad de Amazon EKS Pod".

**Crea trust-relationship.json**:

Crea trust-relationship.json para permitir que EKS Service Principal asuma este rol para Pod Identity. Luego, crea un rol con esta política de confianza:

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

**archivo trust-relationship.json**:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

**Adjunta la política de rol al rol IAM**:

Adjunta la política de rol del paso anterior al rol IAM que creaste:

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

**Crea una asociación de identidad de pod**:

Crea una asociación de identidad de pod entre el rol IAM y la cuenta de servicio de Trident (trident-controller)

```
aws eks create-pod-identity-association \
  --cluster-name <EKS_CLUSTER_NAME> \
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \
  --namespace trident --service-account trident-controller
```

**Función IAM para la asociación de cuenta de servicio (IRSA)**

**Usando la AWS CLI**:

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

**Archivo trust-relationship.json:**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

Actualiza los siguientes valores en el archivo `trust-relationship.json`:

- **<account_id>** - Tu ID de cuenta AWS
- **<oidc_provider>** - El OIDC de tu cluster EKS. Puedes obtener el oidc_provider ejecutando:

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

**Adjunta el rol IAM con la política IAM**:

Una vez que el rol ha sido creado, adjunta la política (que fue creada en el paso anterior) al rol usando este comando:

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

**Verifica que el proveedor OICD está asociado**:

Verifica que tu proveedor OIDC está asociado con tu cluster. Puedes verificarlo usando este comando:

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Si la salida está vacía, usa el siguiente comando para asociar IAM OIDC a tu clúster:

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

**Si usas eksctl**, usa el siguiente ejemplo para crear una función IAM para la cuenta de servicio en EKS:

```
eksctl create iamserviceaccount --name trident-controller --namespace
trident \
  --cluster <my-cluster> --role-name AmazonEKS_FSxN_CSI_DriverRole
--role-only \
  --attach-policy-arn <IAM-Policy ARN> --approve
```

#### Instala Trident

Trident agiliza la administración del almacenamiento de Amazon FSx para NetApp ONTAP en Kubernetes para que tus desarrolladores y administradores se enfoquen en la puesta en marcha de aplicaciones. Puedes instalar Trident usando uno de los siguientes métodos:

- Helm
- Complemento EKS

Si quieres hacer uso de la funcionalidad de instantáneas, instala el complemento controlador de instantáneas CSI. Consulta "Habilita la funcionalidad de instantáneas para volúmenes CSI" para más información.

##### Instala Trident con helm

**Identidad de pod**

1. Agrega el repositorio Trident Helm:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. Instala Trident usando el siguiente ejemplo:

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 --namespace trident --create-namespace
```

Puedes usar el comando `helm list` para revisar detalles de la instalación como nombre, espacio de nombres, gráfico, estado, versión de la aplicación y número de revisión.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2502.0    25.02.0
```

**Asociación de cuenta de servicio (IRSA)**

1. Agrega el repositorio Trident Helm:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. Establece los valores de **proveedor de nube** e **identidad de nube**:

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

Puedes usar el comando `helm list` para revisar detalles de la instalación como nombre, espacio de nombres, gráfico, estado, versión de la aplicación y número de revisión.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                           APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2510.0    25.10.0
```

**NOTA**

Si estás planeando usar iSCSI, asegúrate de que iSCSI esté habilitado en tu máquina cliente. Si estás usando AL2023 Worker node OS, puedes automatizar la instalación del cliente iSCSI añadiendo el parámetro node prep en la instalación de helm:

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1 --namespace trident --create-namespace --
set nodePrep={iscsi}
```

#### Instala Trident mediante el add-on EKS

El complemento Trident EKS incluye los últimos parches de seguridad, correcciones de errores y está validado por AWS para funcionar con Amazon EKS. El complemento EKS te permite asegurarte de forma constante de que tus clústeres de Amazon EKS sean seguros y estables, y reducir la cantidad de trabajo que necesitas hacer para instalar, configurar y actualizar complementos.

### Prerrequisitos

Asegúrate de tener lo siguiente antes de configurar el complemento Trident para AWS EKS:

- Una cuenta de clúster de Amazon EKS con suscripción de complemento
- Permisos de AWS para AWS marketplace:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Tipo de AMI: Amazon Linux 2 (AL2_x86_64) o Amazon Linux 2 Arm(AL2_ARM_64)
- Tipo de nodo: AMD o ARM
- Un sistema de archivos existente de Amazon FSx para NetApp ONTAP

### Habilita el complemento Trident para AWS

**Consola de gestión**

1. Abre la consola de Amazon EKS en https://console.aws.amazon.com/eks/home#/clusters.
2. En el panel de navegación izquierdo, selecciona **Clusters**.
3. Selecciona el nombre del clúster para el que quieres configurar el complemento NetApp Trident CSI.
4. Selecciona **Add-ons** y luego **Get more add-ons**.
5. Sigue estos pasos para seleccionar el complemento:
   a. Desplázate hasta la sección **AWS Marketplace add-ons** y escribe **"Trident"** en el cuadro de búsqueda.
   b. Selecciona la casilla de comprobación en la esquina superior derecha del cuadro Trident by NetApp.
   c. Selecciona **Siguiente**.
6. En la página de configuración **Configurar add-ons seleccionados**, haz lo siguiente:

   **NOTA** **Omite estos pasos si estás usando la asociación Pod Identity.**

   a. Selecciona la **Version** que te gustaría usar.
   b. Si estás usando la autenticación IRSA, asegúrate de establecer los valores de configuración disponibles en la configuración opcional:
      - Selecciona la **Version** que te gustaría usar.
      - Sigue el **Add-on configuration schema** y establece el parámetro **configurationValues** en la sección **Configuration values** al role-arn que creaste en el paso anterior (el valor debe tener el siguiente formato):

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

   +
   Si seleccionas Override como método de resolución de conflictos, una o más de las configuraciones del add-on existente pueden sobrescribirse con las configuraciones del Amazon EKS add-on. Si no habilitas esta opción y hay un conflicto con tus configuraciones existentes, la operación falla. Puedes usar el mensaje de error resultante para solucionar el conflicto. Antes de seleccionar esta opción, asegúrate de que el Amazon EKS add-on no gestione configuraciones que necesites administrar tú mismo.

7. Elige **Siguiente**.
8. En la página **Revisar y agregar**, elige **Crear**.

   Después de que se complete la instalación del complemento, verás tu complemento instalado.

**AWS CLI**

**1. Crea el `add-on.json` file**:

**Para la identidad del pod, usa el siguiente formato**:

**NOTA** Usa la

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

**Para la autenticación de IRSA, usa el siguiente formato**:

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

**NOTA** Reemplaza `<role ARN>` con el ARN del rol que se creó en el paso anterior.

**2. Instala el complemento Trident EKS.**

```
aws eks create-addon --cli-input-json file://add-on.json
```

**eksctl**

El siguiente comando de ejemplo instala el complemento Trident EKS:

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

**Actualiza el add-on Trident EKS**

**Consola de gestión**

1. Abre la consola de Amazon EKS https://console.aws.amazon.com/eks/home#/clusters.
2. En el panel de navegación izquierdo, selecciona **Clusters**.
3. Selecciona el nombre del clúster para el que quieres actualizar el software complementario Trident CSI de NetApp.
4. Selecciona la pestaña **Add-ons**.
5. Selecciona **Trident by NetApp** y luego selecciona **Editar**.
6. En la página **Configura Trident por NetApp**, haz lo siguiente:
   a. Selecciona la **Version** que te gustaría usar.
   b. Expande las **Configuraciones de configuración opcionales** y modifícalas según sea necesario.
   c. Selecciona **Guardar cambios**.

**AWS CLI**

El siguiente ejemplo actualiza el add-on EKS:

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

**eksctl**

- Comprueba la versión actual de tu FSxN Trident CSI add-on. Reemplaza `my-cluster` con el nombre de tu clúster.

  ```
  eksctl get addon --name netapp_trident-operator --cluster my-cluster
  ```

  **Ejemplo de salida:**

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v25.6.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- Actualiza el software complementario a la versión que aparece bajo UPDATE AVAILABLE en la salida del paso anterior.

  ```
  eksctl update addon --name netapp_trident-operator --version
  v25.6.0-eksbuild.1 --cluster my-cluster --force
  ```

Si quitas la opción `--force` y alguna de las configuraciones del complemento de Amazon EKS entra en conflicto con tus configuraciones actuales, entonces la actualización del complemento de Amazon EKS falla; recibes un mensaje de error para ayudarte a resolver el conflicto. Antes de especificar esta opción, asegúrate de que el complemento de Amazon EKS no administre configuraciones que necesites administrar, porque esas configuraciones se sobrescriben con esta opción. Para más información sobre otras opciones para esta configuración, consulta "Complementos". Para más información sobre la administración de campos de Kubernetes de Amazon EKS, consulta "Gestión de campos de Kubernetes".

### Desinstala/elimina el complemento Trident EKS

Tienes dos opciones para eliminar un complemento de Amazon EKS:

- **Conservar el software complementario en tu clúster** – Esta opción elimina la administración de cualquier configuración por parte de Amazon EKS. También elimina la capacidad de Amazon EKS para notificarte sobre actualizaciones y actualizar automáticamente el Amazon EKS add-on después de que inicies una actualización. Sin embargo, conserva el software complementario en tu clúster. Esta opción hace que el add-on sea una instalación autogestionada, en lugar de un Amazon EKS add-on. Con esta opción, no hay tiempo de inactividad para el add-on. Retén la `--preserve` opción en el comando para conservar el add-on.
- **Elimina el software complementario por completo de tu clúster** – NetApp recomienda que elimines el complemento de Amazon EKS de tu clúster solo si no hay recursos en tu clúster que dependan de él. Elimina la opción `--preserve` del comando `delete` para eliminar el complemento.

**NOTA** Si el software complementario tiene una cuenta IAM asociada, la cuenta IAM no se elimina.

**Consola de gestión**

1. Abre la consola de Amazon EKS en https://console.aws.amazon.com/eks/home#/clusters.
2. En el panel de navegación izquierdo, selecciona **Clusters**.
3. Selecciona el nombre del clúster del que quieres quitar el NetApp Trident CSI add-on.
4. Selecciona la pestaña **Complementos** y luego selecciona **Trident by NetApp**.*
5. Selecciona **Eliminar**.
6. En el cuadro de diálogo **Remove netapp_trident-operator confirmation**, haz lo siguiente:
   a. Si quieres que Amazon EKS deje de administrar la configuración del software complementario, selecciona **Conservar en el clúster**. Haz esto si quieres conservar el software complementario en tu clúster para que puedas administrar tú mismo todas las configuraciones del complemento.
   b. Ingresa **netapp_trident-operator**.
   c. Selecciona **Eliminar**.

**AWS CLI**

Reemplaza `my-cluster` con el nombre de tu clúster y luego ejecuta el siguiente comando.

```
aws eks delete-addon --cluster-name my-cluster --addon-name
netapp_trident-operator --preserve
```

**eksctl**

El siguiente comando desinstala el complemento Trident EKS:

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

### Configurar una clase de almacenamiento

El "Objeto de Kubernetes StorageClass" identifica a un proveedor y le indica cómo aprovisionar volúmenes. Esta sección te muestra cómo configurar un objeto de Kubernetes StorageClass que especifica a Trident como el proveedor.

#### Crear un objeto StorageClass

Cuando creas un StorageClass para FSx for ONTAP, Trident creará automáticamente la configuración del backend.

| | |
|---|---|
| **NOTA** | Si quieres configurar manualmente el backend de almacenamiento, consulta la sección [create-a-kubernetes-storageclass-without-automatic-backend-configuration] para saber cómo crear el backend Trident y la clase de almacenamiento por separado. |

#### Especifica los parámetros requeridos StorageClass

Los siguientes tres parámetros deben definirse al crear un StorageClass:

| Parámetro | Requerido | Tipo | Descripción |
| --- | --- | --- | --- |
| `fsxFilesystemID` | Sí | cadena | FSx para NetApp ONTAP filesystem ID |
| `storageDriverName` | Sí | cadena | Controlador de almacenamiento Trident (por ejemplo, `ontap-nas` o `ontap-san`) |
| `credentialsName` | Sí | cadena | Nombre del secreto de Kubernetes que contiene FSx for ONTAP credenciales |

#### Especifica parámetros opcionales

Puedes pasar parámetros opcionales de backend a través de StorageClass. Define todos los valores opcionales como cadenas en la sección StorageClass `parameters`. Para obtener una lista completa de los parámetros de backend, consulta: "Configuración del backend de FSx para NetApp ONTAP".

#### Ejemplo de archivos de configuración de StorageClass.

El siguiente ejemplo muestra un StorageClass que activa la configuración automática del backend.

**YAML**

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-fsx-demo
  annotations:
    description: "Demo StorageClass for FSx for NetApp ONTAP"
provisioner: csi.trident.netapp.io
parameters:
  fsxFilesystemID: "fs-0abc123"
  storageDriverName: "ontap-nas"
  credentialsName: trident-fsx-credentials
allowVolumeExpansion: true
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

**JSON**

```json
{
  "apiVersion": "storage.k8s.io/v1",
  "kind": "StorageClass",
  "metadata": {
    "name": "ontap-fsx-demo",
    "annotations": {
      "description": "Demo StorageClass for FSx for NetApp ONTAP"
    }
  },
  "provisioner": "csi.trident.netapp.io",
  "parameters": {
    "fsxFilesystemID": "fs-0abc123",
    "storageDriverName": "ontap-nas",
    "credentialsName": "trident-fsx-credentials"
  },
  "allowVolumeExpansion": true,
  "reclaimPolicy": "Delete",
  "volumeBindingMode": "Immediate"
}
```

### Crear el StorageClass

Una vez que hayas creado tu archivo de configuración, ejecuta el siguiente comando para crear la clase de almacenamiento.

```
kubectl create -f storage-class-ontapnas.yaml
```

Ahora deberías ver una clase de almacenamiento **basic-csi** tanto en Kubernetes como en Trident, y Trident debería haber descubierto los pools en el backend.

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

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

Después de aplicar StorageClass, Trident crea el backend automáticamente. Luego, puedes crear PersistentVolumeClaims que hacen referencia a este StorageClass.

#### Verifica el estado de la configuración del backend

Trident registra el resultado de la creación del backend en las anotaciones de StorageClass.

| Anotación | Descripción |
|---|---|
| `trident.netapp.io/configuratorStatus` | Resultado de la configuración (`Success` o `Failure`) |
| `trident.netapp.io/configuratorMessage` | Estado detallado o mensaje de error |
| `trident.netapp.io/configuratorName` | Nombre del recurso interno del configurador |
| `trident.netapp.io/managed` | Indica que el StorageClass está gestionado por Trident |
| `trident.netapp.io/additionalStoragePools` | Grupos de almacenamiento creados para este backend |

Para verificar el estado, ejecuta:

```
kubectl get storageclass ontap-fsx-demo -o yaml
```

Confirma que `trident.netapp.io/configuratorStatus` está configurado en `Success`. Si el valor es `Failure`, revisa `trident.netapp.io/configuratorMessage` para ver el error.

#### Añade sistemas de archivos FSxN adicionales

Si necesitas capacidad de almacenamiento adicional mientras sigues usando el mismo StorageClass, añade ID de sistemas de archivos FSxN adicionales.

Edita el StorageClass y añade la siguiente anotación:

```
metadata:
  annotations:
    trident.netapp.io/additionalFsxNFileSystemID: '["fs-
xxxxxxxxxxxxxxxxx"]'
```

Después de aplicar el cambio, Trident actualiza la configuración del backend y actualiza las anotaciones de StorageClass.

### Consideraciones operativas y limitaciones

- Eliminar un StorageClass que tiene la configuración automática de backend normalmente elimina el backend Trident asociado. Esto puede interrumpir la conectividad de almacenamiento y afectar las cargas de trabajo en ejecución. Valida el impacto antes de eliminar un StorageClass administrado.
- La configuración automática del backend solo es compatible con AWS FSx for NetApp ONTAP.

### Crear un Kubernetes StorageClass sin configuración automática del backend

Si quieres crear el backend de Trident y StorageClass por separado, entonces sigue estos pasos.

### Entiende cómo funciona la configuración automática del backend

Trident obtiene la configuración del backend a partir de la definición de StorageClass. Cuando aplicas la StorageClass, Trident valida los parámetros necesarios, crea el backend y anota la StorageClass con el estado.

Trident crea el VolumeSnapshotClass solo una vez. Trident reutiliza el mismo VolumeSnapshotClass para los siguientes StorageClasses.

### Crear el backend Trident

Para crear un backend Trident, necesitas crear un archivo de configuración en formato JSON o YAML. El archivo debe especificar el tipo de almacenamiento que quieres (NAS o SAN), el sistema de archivos, la SVM de la que lo vas a obtener y cómo autenticarte con ella. El siguiente ejemplo muestra cómo definir almacenamiento basado en NAS y usar un secreto de AWS para guardar las credenciales de la SVM que quieres usar:

**YAML**

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

**JSON**

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
    "namespace": "trident"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFilesystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

### Detalles del controlador de FSx for ONTAP

Puedes integrar Trident con Amazon FSx for NetApp ONTAP usando los siguientes controladores:

| Nombre del controlador | Descripción |
|---|---|
| `ontap-san` | Cada PV aprovisionado es un LUN dentro de su propio volumen de Amazon FSx for NetApp ONTAP. Recomendado para almacenamiento en bloque. |
| `ontap-nas` | Cada PV aprovisionado es un volumen completo de Amazon FSx for NetApp ONTAP. Recomendado para NFS y SMB. |
| `ontap-san-economy` | Cada PV aprovisionado es un LUN con un número configurable de LUNs por volumen de Amazon FSx para NetApp ONTAP. |
| `ontap-nas-economy` | Cada PV aprovisionado es un qtree, con un número configurable de qtrees por volumen de Amazon FSx para NetApp ONTAP. |
| `ontap-nas-flexgroup` | Cada PV aprovisionado es un volumen completo de Amazon FSx for NetApp ONTAP FlexGroup. |

Para obtener detalles del controlador, consulta "Controladores NAS" y "Controladores SAN".

### Crea el backend

Después de crear el archivo de configuración, ejecuta los siguientes comandos para crear y validar la configuración del backend de Trident (TBC):

- Crea la configuración del backend de Trident (TBC) desde el archivo yaml y ejecuta el siguiente comando:

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- Valida que la configuración de backend de Trident (TBC) se creó correctamente:

```
Kubectl get tbc -n trident
```

```
NAME                          BACKEND NAME          BACKEND UUID
PHASE   STATUS

backend-tbc-ontap-nas         tbc-ontap-nas         933e0071-66ce-4324-
b9ff-f96d916ac5e9   Bound   Success
```

Para más información sobre otras opciones de configuración, consulta la sección [Backend-advanced-configuration-and-examples] a continuación.

### Configura una clase de almacenamiento sin configuración automática del backend

A continuación se muestran ejemplos de configuraciones de Storage Class para su uso con Trident y FSx for ONTAP.

#### Clase de almacenamiento para NFS

Puedes usar este ejemplo para configurar StorageClass para volúmenes que usan NFS (consulta la sección de atributos de Trident a continuación para la lista completa de atributos):

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true"
```

#### Storage Calss para iSCSI

Usa este ejemplo para configurar StorageClass para volúmenes que usan iSCSI:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  provisioningType: "thin"
  snapshots: "true"
```

#### Clase de almacenamiento con NFSv3 y AWS Bottlerocket

Para aprovisionar volúmenes NFSv3 en AWS Bottlerocket, agrega lo requerido `mountOptions` a la clase de almacenamiento:

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

### Atributos de Trident StorageClass

Estos parámetros determinan qué grupos de almacenamiento administrados por Trident se deben utilizar para aprovisionar volúmenes de un tipo dado.

| Atributo | Tipo | Valores | Oferta | Solicitud | Con el apoyo de |
|---|---|---|---|---|---|
| medios[1] | cadena | hdd, híbrido, ssd | El pool contiene medios de este tipo; híbrido significa ambos | Tipo de medio especificado | ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san |
| provisioningType | cadena | delgado, grueso | Pool admite este método de aprovisionamiento | Método de aprovisionamiento especificado | grueso: todo ontap; fino: todo ontap & solidfire-san |
| backendType | cadena | ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy | Pool pertenece a este tipo de backend | Backend especificado | Todos los drivers |
| instantáneas | bool | verdadero, falso | El pool admite volúmenes con instantáneas | Volumen con instantáneas habilitadas | ontap-nas, ontap-san, solidfire-san |
| clones | bool | verdadero, falso | El pool admite la clonación de volúmenes | Volumen con clones habilitados | ontap-nas, ontap-san, solidfire-san |

| Atributo | Tipo | Valores | Oferta | Solicitud | Con el apoyo de |
|---|---|---|---|---|---|
| cifrado | bool | verdadero, falso | El pool admite volúmenes cifrados | Volumen con cifrado habilitado | ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san |
| IOPS | entero | entero positivo | El pool es capaz de garantizar IOPS en este rango | Volumen garantizó estas IOPS | solidfire-san |

[1]: No soportado por ONTAP Select o FSx for ONTAP systems

Consulta "Objetos de Kubernetes y Trident" para ver detalles sobre cómo las clases de almacenamiento interactúan con el `PersistentVolumeClaim` y los parámetros para controlar cómo Trident aprovisiona volúmenes.

### Crear la clase de almacenamiento

Una vez que hayas configurado StorageClass, puedes crearlo en Kubernetes.

**Pasos**

1. Este es un objeto de Kubernetes, así que usa `kubectl` para crearlo en Kubernetes.

```
kubectl create -f storage-class-ontapnas.yaml
```

2. Ahora deberías ver una clase de almacenamiento **basic-csi** tanto en Kubernetes como en Trident, y Trident debería haber descubierto los pools en el backend.

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

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

#### Aprovisionar volúmenes SMB

Puedes aprovisionar volúmenes SMB usando el controlador `ontap-nas`. Sin embargo, para hacerlo debes completar estos pasos: "Prepárate para aprovisionar volúmenes SMB".

#### Configuración avanzada de backend y ejemplos

Consulta la siguiente tabla para ver las opciones de configuración del backend:

| Parámetro | Descripción | Ejemplo |
| --- | --- | --- |
| `version` | | Siempre 1 |
| `storageDriverName` | Nombre del controlador de almacenamiento | `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy` |
| `backendName` | Nombre personalizado o el backend de almacenamiento | Nombre del driver + "_" + dataLIF |
| `managementLIF` | Dirección IP de un clúster o de una LIF de administración de SVM. Se puede especificar un nombre de dominio completo (FQDN). Se puede configurar para usar direcciones IPv6 si Trident se instaló usando el flag de IPv6. Las direcciones IPv6 deben definirse entre corchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Si proporcionas el `fsxFilesystemID` bajo el campo `aws`, no necesitas proporcionar el `managementLIF` porque Trident obtiene la información de SVM `managementLIF` desde AWS. Así que debes proporcionar credenciales para un usuario bajo la SVM (por ejemplo: vsadmin) y el usuario debe tener el rol `vsadmin`. | "10.0.0.1", "[2001:1234:abcd::fefe]" |

| Parámetro | Descripción | Ejemplo |
| --- | --- | --- |
| `dataLIF` | Dirección IP del LIF de protocolo. **Controladores NAS de ONTAP**: NetApp recomienda especificar dataLIF. Si no se proporciona, Trident obtiene los dataLIF del SVM. Puedes especificar un nombre de dominio completo (FQDN) para las operaciones de montaje NFS, lo que te permite crear un DNS de round-robin para balancear la carga entre varios dataLIF. Se puede cambiar después de la configuración inicial. **Controladores SAN de ONTAP**: No lo especifiques para iSCSI. Trident usa ONTAP Selective LUN Map para descubrir los LIF de iSCSI necesarios para establecer una sesión multipath. Se genera una advertencia si dataLIF se define explícitamente. Se puede configurar para usar direcciones IPv6 si Trident se instaló usando el flag de IPv6. Las direcciones IPv6 deben definirse entre corchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. | |
| `autoExportPolicy` | Habilita la creación y actualización automática de políticas de exportación [Boolean]. Usando las opciones `autoExportPolicy` y `autoExportCIDRs`, Trident puede gestionar las políticas de exportación automáticamente. | `false` |
| `autoExportCIDRs` | Lista de CIDR para filtrar las direcciones IP de los nodos de Kubernetes cuando `autoExportPolicy` está habilitado. Usando las opciones `autoExportPolicy` y `autoExportCIDRs`, Trident puede gestionar las políticas de exportación automáticamente. | "["0.0.0.0/0", "::/0"]" |
| `labels` | Conjunto de etiquetas arbitrarias con formato JSON para aplicar en volúmenes | "" |
| `clientCertificate` | Valor codificado en Base64 del certificado del cliente. Usado para auth basada en certificados | "" |

| Parámetro | Descripción | Ejemplo |
| --- | --- | --- |
| `clientPrivateKey` | Valor codificado en Base64 de la clave privada del cliente. Usado para auth basada en certificados | "" |
| `trustedCACertificate` | Valor codificado en Base64 del certificado de CA de confianza. Opcional. Se usa para la autenticación basada en certificados. | "" |
| `username` | Nombre de usuario para conectarte al clúster o SVM. Se utiliza para la autenticación basada en credenciales. Por ejemplo, vsadmin. | |
| `password` | Contraseña para conectarse al clúster o SVM. Se utiliza para la autenticación basada en credenciales. | |
| `svm` | Máquina virtual de almacenamiento que vas a usar | Derivado si se especifica un LIF de gestión de SVM. |
| `storagePrefix` | Prefijo utilizado al aprovisionar nuevos volúmenes en la SVM. No se puede modificar después de la creación. Para actualizar este parámetro, tendrás que crear un nuevo backend. | `trident` |
| `limitAggregateUsage` | **No especifiques para Amazon FSx for NetApp ONTAP.** Los `fsxadmin` y `vsadmin` no tienen los permisos necesarios para recuperar el uso agregado y limitarlo usando Trident. | No uses. |
| `limitVolumeSize` | Falla el aprovisionamiento si el tamaño del volumen solicitado supera este valor. También restringe el tamaño máximo de los volúmenes que administra para qtrees y LUNs, y la `qtreesPerFlexvol` opción permite personalizar el número máximo de qtrees por FlexVol volume | "" (no aplicado por defecto) |
| `lunsPerFlexvol` | LUNs máximos por FlexVol volume, deben estar en el rango [50, 200]. Solo para SAN. | “100” |

| Parámetro | Descripción | Ejemplo |
| --- | --- | --- |
| `debugTraceFlags` | Indicadores de depuración para utilizar cuando estés solucionando problemas. Por ejemplo, {"api":false, "method":true} no uses `debugTraceFlags` a menos que estés resolviendo problemas y necesites un volcado de registro detallado. | null |
| `nfsMountOptions` | Lista de opciones de montaje NFS separadas por comas. Las opciones de montaje para volúmenes persistentes de Kubernetes normalmente se especifican en las clases de almacenamiento, pero si no se especifican opciones de montaje en una clase de almacenamiento, Trident usará las opciones de montaje especificadas en el archivo de configuración del backend de almacenamiento. Si no se especifican opciones de montaje en la clase de almacenamiento ni en el archivo de configuración, Trident no establecerá ninguna opción de montaje en un volumen persistente asociado. | "" |
| `nasType` | Configura la creación de volúmenes NFS o SMB. Las opciones son `nfs`, `smb` o null. **Debes establecerlo en `smb` para volúmenes SMB.** Si lo configuras en null, se usarán volúmenes NFS por defecto. | `nfs` |
| `qtreesPerFlexvol` | Máximo de qtrees por FlexVol volume, debe estar en el rango [50, 300] | `"200"` |
| `smbShare` | Puedes especificar uno de los siguientes: el nombre de un recurso compartido SMB creado usando Microsoft Management Console o ONTAP CLI, o un nombre para permitir que Trident cree el recurso compartido SMB. Este parámetro es obligatorio para los backends de Amazon FSx for ONTAP. | `smb-share` |

| Parámetro | Descripción | Ejemplo |
|---|---|---|
| `useREST` | Parámetro booleano para usar las ONTAP REST APIs. Cuando se establece en `true`, Trident usará las ONTAP REST APIs para comunicarse con el backend. Esta función requiere ONTAP 9.11.1 y versiones posteriores. Además, el rol de inicio de sesión de ONTAP utilizado debe tener acceso a la aplicación `ontap`. Esto se cumple con los roles predefinidos `vsadmin` y `cluster-admin`. | `false` |
| `aws` | Puedes especificar lo siguiente en el archivo de configuración para AWS FSx for ONTAP: - `fsxFilesystemID`: especifica el ID del sistema de archivos de AWS FSx. - `apiRegion`: nombre de la región de la API de AWS. - `apikey`: clave de API de AWS. - `secretKey`: clave secreta de AWS. | `""`<br>`""`<br>`""` |
| `credentials` | Especifica las credenciales de FSx SVM que se van a guardar en AWS Secrets Manager. - `name`: Amazon Resource Name (ARN) del secreto, que contiene las credenciales de SVM. - `type`: Establécelo en `awsarn`. Consulta "Crea un secreto de AWS Secrets Manager" para más información. | |

#### Opciones de configuración de backend para aprovisionar volúmenes

Puedes controlar el aprovisionamiento predeterminado usando estas opciones en la `defaults` sección de la configuración. Por ejemplo, mira los ejemplos de configuración abajo.

| Parámetro | Descripción | Predeterminado |
|---|---|---|
| `spaceAllocation` | Asignación de espacio para LUNs | `true` |
| `spaceReserve` | Modo de reserva de espacio; "ninguno" (fino) o "volumen" (grueso) | `none` |
| `snapshotPolicy` | Política de SnapVault a utilizar | `none` |

| Parámetro | Descripción | Predeterminado |
| --- | --- | --- |
| `qosPolicy` | Grupo de políticas de QoS para asignar a los volúmenes creados. Elige una de qosPolicy o adaptiveQosPolicy por pool de almacenamiento o backend. Usar grupos de políticas de QoS con Trident requiere ONTAP 9.8 o una versión posterior. Deberías usar un grupo de políticas de QoS no compartido y asegurarte de que el grupo de políticas se aplique a cada componente individualmente. Un grupo de políticas de QoS compartido impone el límite máximo para el rendimiento total de todas las cargas de trabajo. | "" |
| `adaptiveQosPolicy` | Grupo de políticas de QoS adaptativo para asignar a los volúmenes creados. Elige una de qosPolicy o adaptiveQosPolicy por pool de almacenamiento o backend. No compatible con ontap-nas-economy. | "" |
| `snapshotReserve` | Porcentaje de volumen reservado para snapshots "0" | Si `snapshotPolicy` es `none`, `else` " |
| `splitOnClone` | Divide un clon de su padre al momento de su creación | `false` |
| `encryption` | Habilita NetApp Volume Encryption (NVE) en el nuevo volumen; el valor predeterminado es `false`. NVE debe tener licencia y estar habilitado en el clúster para usar esta opción. Si NAE está habilitado en el backend, cualquier volumen aprovisionado en Trident tendrá NAE habilitado. Para más información, consulta: "Cómo funciona Trident con NVE y NAE". | `false` |
| `luksEncryption` | Activa el cifrado LUKS. Consulta "Usa Linux Unified Key Setup (LUKS)". Solo para SAN. | "" |
| `tieringPolicy` | Política de tiering a usar `none` | |
| `unixPermissions` | Modo para nuevos volúmenes. **Dejar vacío para volúmenes SMB.** | "" |

| Parámetro | Descripción | Predeterminado |
| --- | --- | --- |
| `securityStyle` | Estilo de seguridad para nuevos volúmenes. NFS admite `mixed` y `unix` estilos de seguridad. SMB admite `mixed` y `ntfs` estilos de seguridad. | NFS predeterminado es `unix`. SMB predeterminado es `ntfs`. |

### Configura un PVC

Esta sección incluye instrucciones sobre cómo crear un PersistentVolumeClaim (PVC) que utiliza la StorageClass de Kubernetes configurada para solicitar un PV. Si tiene éxito, luego puedes montar el PV en un pod.

#### Crea el PVC

Una "*PersistentVolumeClaim*" (PVC) es una solicitud de acceso al PersistentVolume en el clúster. El PVC se puede configurar para solicitar almacenamiento de un tamaño o modo de acceso determinados. Usando el StorageClass asociado, el administrador del clúster puede controlar más que solo el tamaño y el modo de acceso de PersistentVolume, como el rendimiento o el nivel de servicio.

Después de crear el backend Trident y StorageClass puedes crear un PVC. Después de crear el PVC, puedes montar el volumen en un pod.

#### Manifiestos de muestra

Los siguientes ejemplos muestran las opciones básicas de configuración del PVC.

**PVC con acceso RWX**

Este ejemplo muestra un PVC básico con acceso RWX que está asociado con una StorageClass llamada `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-gold
```

**Ejemplo de PVC usando iSCSI**

Este ejemplo muestra un PVC básico para iSCSI con acceso RWO que está asociado con un StorageClass llamado `protection-gold`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: protection-gold
```

#### Crear PVC

**Pasos**

1. Crea el PVC.

```
kubectl create -f pvc.yaml
```

2. Verifica el estado del PVC.

```
kubectl get pvc
```

```
NAME        STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage Bound  pv-name 2Gi      RWO                          5m
```

Consulta "Objetos de Kubernetes y Trident" para ver detalles sobre cómo las clases de almacenamiento interactúan con el `PersistentVolumeClaim` y los parámetros para controlar cómo Trident aprovisiona volúmenes.

#### Implementa una aplicación

Una vez creada la clase de almacenamiento y el PVC, puedes montar el PV en un pod. Esta sección muestra el comando de ejemplo y la configuración para conectar el PV a un pod.

#### Implementa una aplicación de ejemplo

**Pasos**

1. Monta el volumen en un pod.

```
kubectl create -f pv-pod.yaml
```

Estos ejemplos muestran configuraciones básicas para conectar el PVC a un pod: **configuración básica**:

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

**NOTA** Puedes monitorear el progreso usando `kubectl get pod --watch`.

2. Verifica que el volumen esté montado en `/my/mount/path`.

```
kubectl exec -it pv-pod -- df -h /my/mount/path
```

```
Filesystem                                                      Size
Used  Avail Use%  Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06  1.1G
320K  1.0G   1%   /my/mount/path
```

Ahora puedes eliminar el Pod. La aplicación del Pod ya no existirá, pero el volumen permanecerá.

```
kubectl delete pod pv-pod
```

#### Configura el complemento Trident EKS en un clúster EKS

NetApp Trident optimiza la gestión del almacenamiento de Amazon FSx for NetApp ONTAP en Kubernetes para que tus desarrolladores y administradores puedan centrarse en la puesta en marcha de aplicaciones. El complemento NetApp Trident EKS incluye los

últimos parches de seguridad, correcciones de errores y está validado por AWS para funcionar con Amazon EKS. El complemento EKS te permite asegurarte de forma constante de que tus clústeres de Amazon EKS sean seguros y estables, y reducir la cantidad de trabajo que necesitas hacer para instalar, configurar y actualizar complementos.

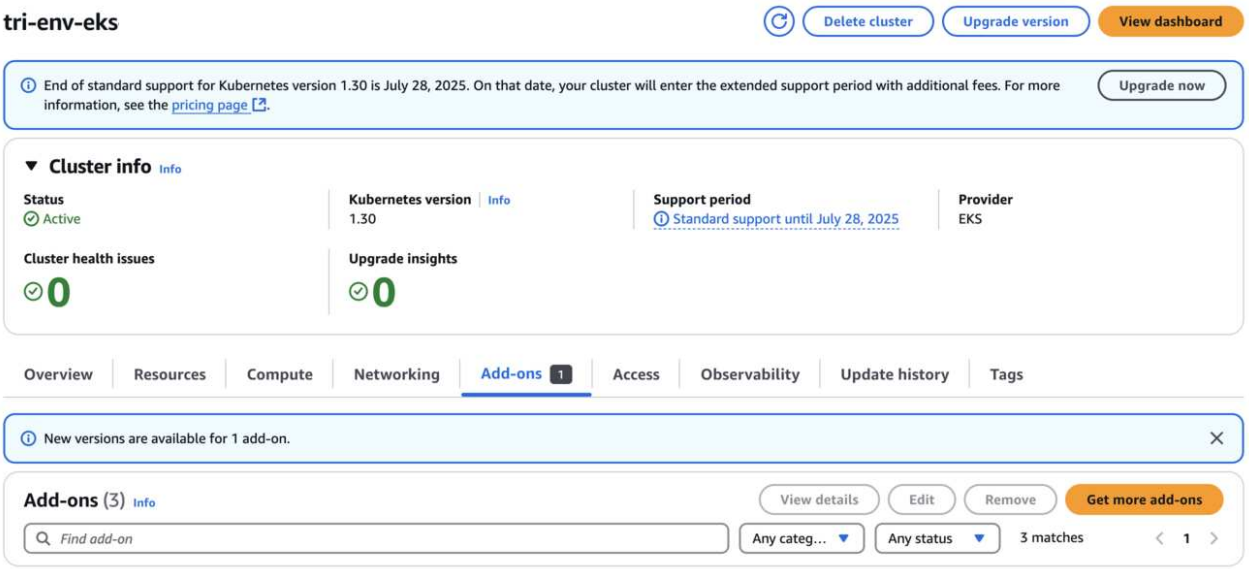
**Prerrequisitos**

Asegúrate de tener lo siguiente antes de configurar el complemento Trident para AWS EKS:

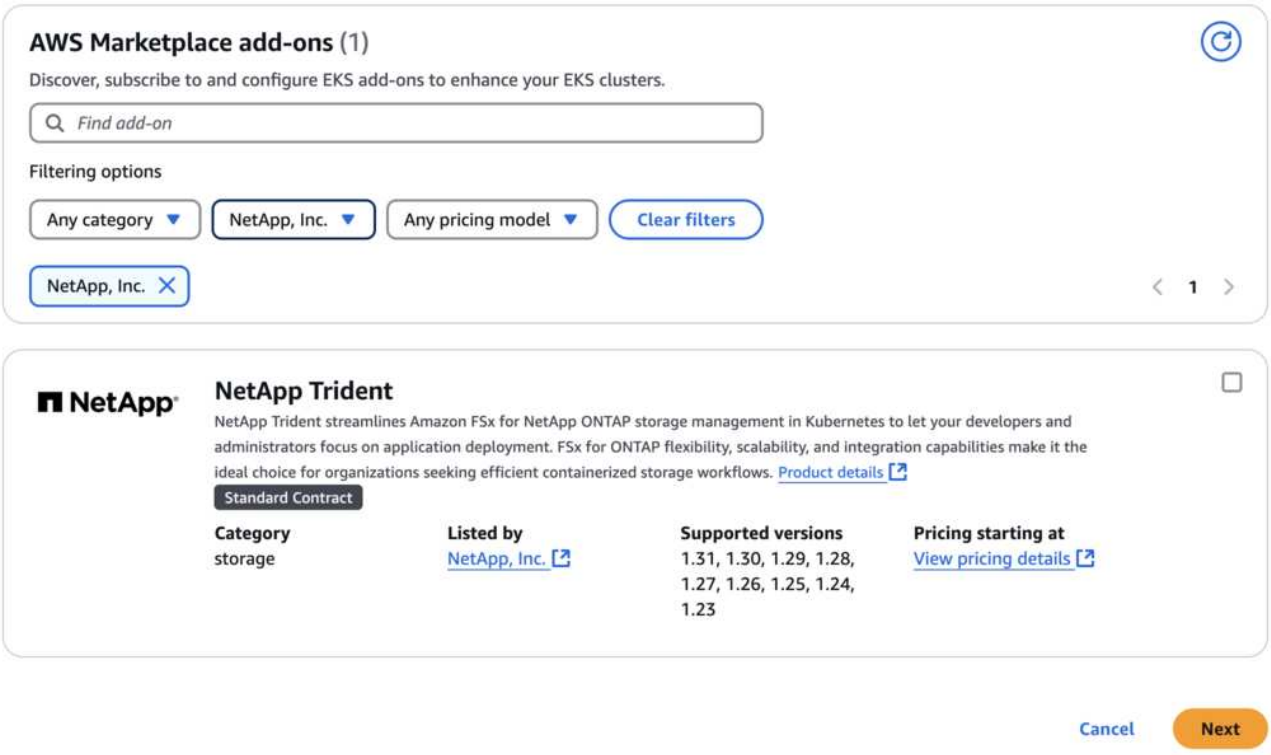
- Una cuenta de clúster de Amazon EKS con permisos para trabajar con complementos. Consulta "Complementos de Amazon EKS".
- Permisos de AWS para AWS marketplace:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Tipo de AMI: Amazon Linux 2 (AL2_x86_64) o Amazon Linux 2 Arm(AL2_ARM_64)
- Tipo de nodo: AMD o ARM
- Un sistema de archivos existente de Amazon FSx para NetApp ONTAP

**Pasos**

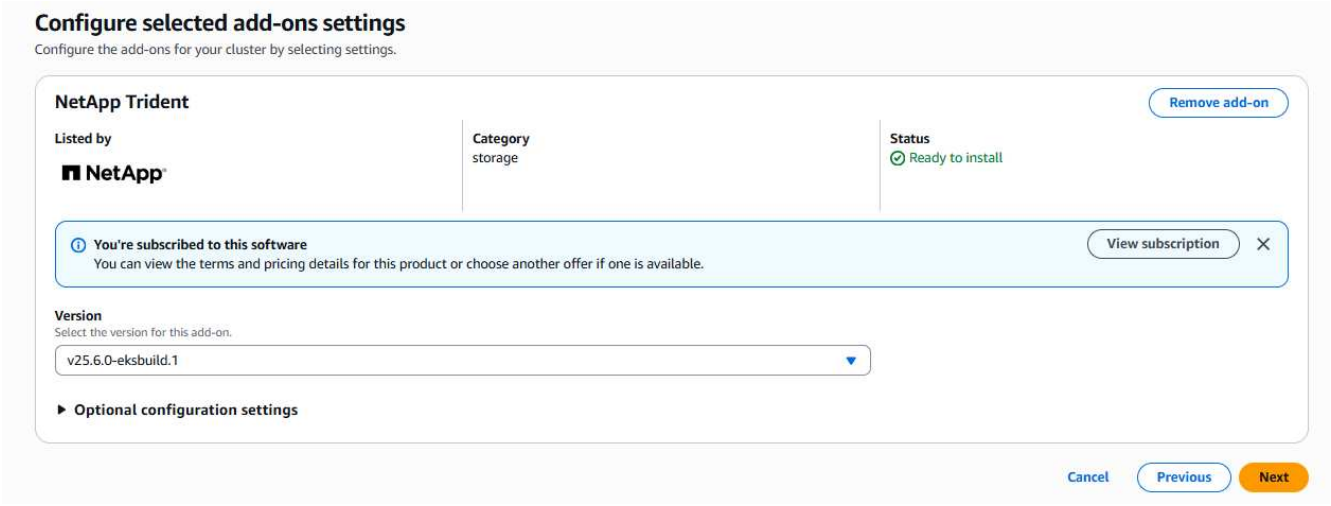
1. Asegúrate de crear un rol de IAM y un secreto de AWS para que los pods de EKS puedan acceder a los recursos de AWS. Para ver las instrucciones, consulta "Crea un rol de IAM y un secreto de AWS".
2. En tu clúster EKS Kubernetes, ve a la pestaña **Add-ons**.



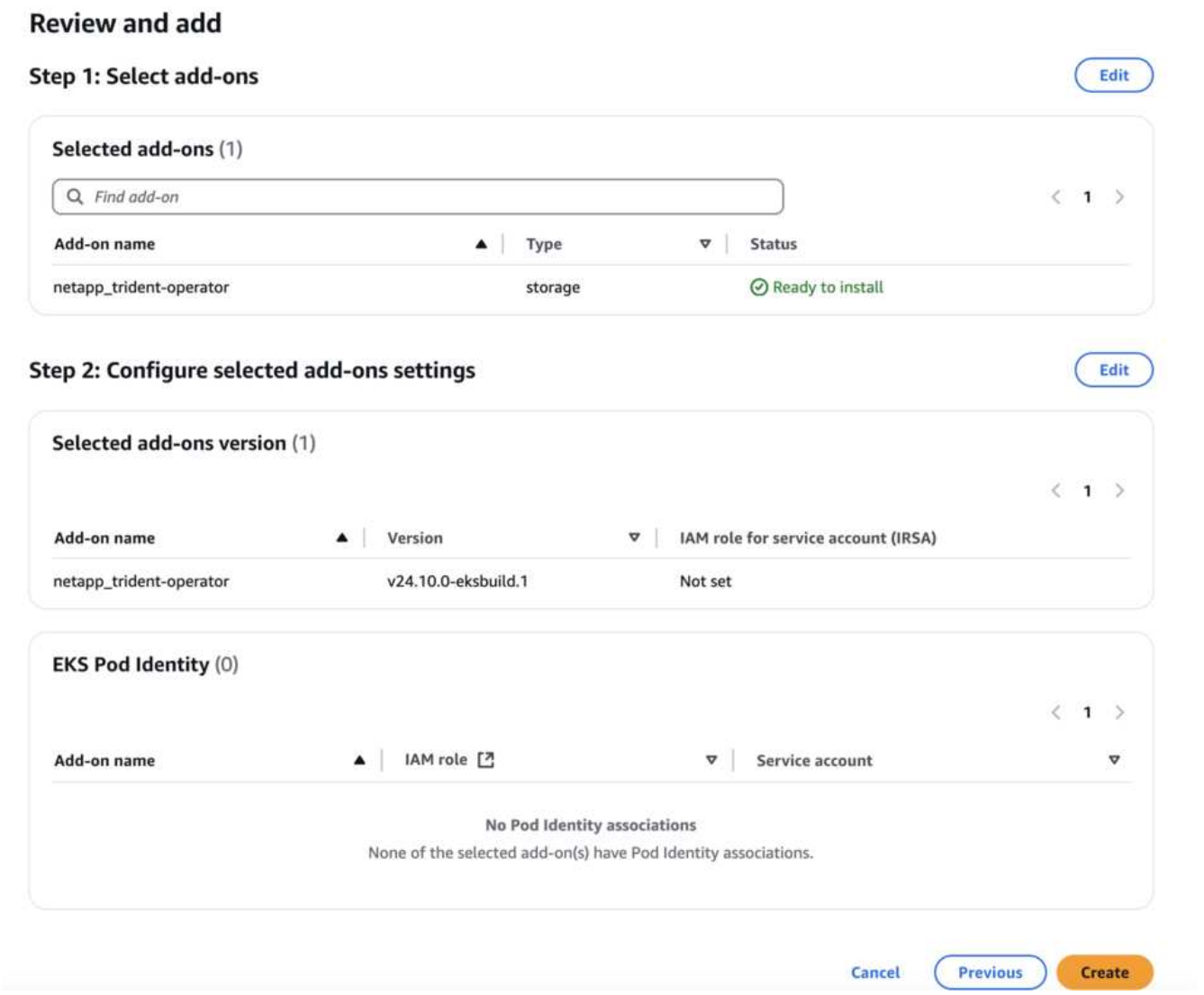

3. Ve a **AWS Marketplace add-ons** y elige la categoría *storage*.



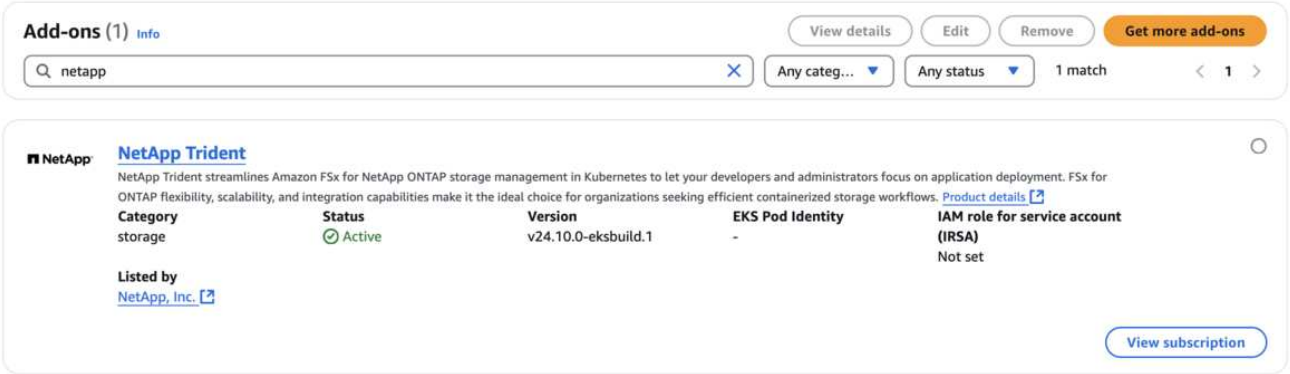

4. Localiza **NetApp Trident** y selecciona la casilla para el complemento Trident, y haz clic en **Siguiente**.
5. Elige la versión deseada del complemento.




6. Configura los ajustes complementarios necesarios.




7. Si estás usando IRSA (IAM roles para service account), consulta los pasos de configuración adicionales "aquí".
8. Selecciona **Crear**.
9. Verifica que el estado del complemento sea *Activo*.




10. Ejecuta el siguiente comando para verificar que Trident está instalado correctamente en el clúster:

```
kubectl get pods -n trident
```

11. Continúa con la configuración y configura el backend de almacenamiento. Para más información, consulta "Configura el backend de almacenamiento".

#### Instala o desinstala el complemento Trident EKS usando la CLI

**Instala el complemento Trident EKS de NetApp usando la CLI:**

El siguiente comando de ejemplo instala el complemento Trident EKS:
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.0-eksbuild.1` (con una versión dedicada)

El siguiente comando de ejemplo instala el complemento Trident EKS versión 25.6.1:
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.1-eksbuild.1` (con una versión dedicada)

El siguiente comando de ejemplo instala el complemento Trident EKS versión 25.6.2:
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.2-eksbuild.1` (con una versión dedicada)

**Desinstala el complemento Trident EKS de NetApp usando la CLI:**

El siguiente comando desinstala el complemento Trident EKS:

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

## Crea backends con kubectl

Un backend define la relación entre Trident y un sistema de almacenamiento. Le dice a Trident cómo comunicarse con ese sistema de almacenamiento y cómo Trident debe aprovisionar volúmenes desde él. Después de instalar Trident, el siguiente paso es crear un backend. La `TridentBackendConfig` Custom Resource Definition (CRD) te permite crear y administrar backends de Trident directamente a través de la interfaz de Kubernetes. Puedes hacer esto usando `kubectl` o la herramienta CLI equivalente para tu distribución de Kubernetes.

### `TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`) es un CRD frontend con espacio de nombres que te permite administrar backends de Trident usando `kubectl`. Los administradores de Kubernetes y almacenamiento ahora pueden crear y administrar backends directamente a través de la CLI de Kubernetes sin necesitar una utilidad de línea de comandos dedicada (`tridentctl`).

Al crear un `TridentBackendConfig` objeto, sucede lo siguiente:

- Trident crea automáticamente un backend según la configuración que proporciones. Esto se representa internamente como un `TridentBackend` (`tbe`, `tridentbackend`) CR.
- El `TridentBackendConfig` está vinculado de forma única a un `TridentBackend` que fue creado por

Trident.

Cada `TridentBackendConfig` mantiene una correspondencia uno a uno con un `TridentBackend`. El primero es la interfaz que se proporciona al usuario para diseñar y configurar backends; el segundo es la forma en que Trident representa el objeto backend real.

| | |
|---|---|
| **ADVERTENCIA** | `TridentBackend`Las CR se crean automáticamente por Trident. No deberías modificarlas. Si quieres actualizar los backends, hazlo modificando el objeto `TridentBackendConfig`. |

Mira el siguiente ejemplo para el formato del `TridentBackendConfig`CR:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

También puedes echar un vistazo a los ejemplos en el directorio "trident-installer" para ver configuraciones de muestra para la plataforma o servicio de almacenamiento que quieras.

El `spec` toma parámetros de configuración específicos del backend. En este ejemplo, el backend usa el `ontap-san` storage driver y usa los parámetros de configuración que se muestran aquí. Para la lista de opciones de configuración para tu storage driver deseado, consulta el "Información de configuración del backend para tu controlador de almacenamiento".

La `spec`sección también incluye `credentials`y `deletionPolicy`campos, que se introdujeron recientemente en el `TridentBackendConfig`CR:

- `credentials`: Este parámetro es obligatorio y contiene las credenciales utilizadas para autenticarse con el sistema/servicio de almacenamiento. Esto se establece como un secreto de Kubernetes creado por el usuario. Las credenciales no se pueden pasar en texto sin formato y generarán un error.
- `deletionPolicy`: Este campo define qué debe suceder cuando se elimina el `TridentBackendConfig`. Puede tomar uno de dos valores posibles:
  - `delete`: Esto elimina tanto `TridentBackendConfig`CR como el backend asociado. Este es el valor predeterminado.
  - `retain`: Cuando se elimina una `TridentBackendConfig` CR, la definición del backend seguirá presente y se puede administrar con `tridentctl`. Configurar la política de eliminación en `retain` permite a los usuarios volver a una versión anterior (pre-21.04) y conservar los backends creados. El valor de este campo se puede actualizar después de que se cree una `TridentBackendConfig`.

| | |
|---|---|
| **NOTA** | El nombre de un backend se establece usando `spec.backendName`. Si no se especifica, el nombre del backend se establece como el nombre del objeto `TridentBackendConfig` (metadata.name). Se recomienda establecer explícitamente los nombres de los backends usando `spec.backendName`. |

| | |
|---|---|
| **CONSEJO** | Los backends que fueron creados con `tridentctl` no tienen un objeto `TridentBackendConfig` asociado. Puedes elegir administrar esos backends con `kubectl` creando un CR `TridentBackendConfig`. Debes tener cuidado de especificar parámetros de configuración idénticos (como `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName` y así sucesivamente). Trident vinculará automáticamente el `TridentBackendConfig` recién creado con el backend preexistente. |

**Resumen de pasos**

Para crear un nuevo backend usando `kubectl`, debes hacer lo siguiente:

1. Crea un "Kubernetes Secret". 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.

Después de crear un backend, puedes observar su estado usando `kubectl get tbc <tbc-name> -n <trident-namespace>` y recopilar detalles adicionales.

**Paso 1: crea un secreto de Kubernetes**

Crea un secreto que contenga las credenciales de acceso para el backend. Esto es único para cada servicio o plataforma de almacenamiento. Aquí tienes un ejemplo:

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

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

Esta tabla resume los campos que deben incluirse en el secreto para cada plataforma de almacenamiento:

| Descripción de los campos secretos de la plataforma de almacenamiento | Secreto | Descripción de los campos |
|---|---|---|
| Azure NetApp Files | clientID | El ID del cliente de un registro de app |
| Element (NetApp HCI/SolidFire) | Punto final | MVIP para el clúster SolidFire con credenciales de inquilino |
| ONTAP | nombre de usuario | Nombre de usuario para conectarse al clúster/SVM. Usada para la autenticación basada en credenciales |
| ONTAP | contraseña | Contraseña para conectarte al clúster/SVM. Usada para la autenticación basada en credenciales |
| ONTAP | clientPrivateKey | Valor codificado en Base64 de la clave privada del cliente. Usado para la autenticación basada en certificados |
| ONTAP | chapUsername | Nombre de usuario de entrada. Obligatorio si useCHAP=true. Para `ontap-san` y `ontap-san-economy` |
| ONTAP | chapInitiatorSecret | Secreto del iniciador de CHAP. Obligatorio si useCHAP=true. Para `ontap-san` y `ontap-san-economy` |
| ONTAP | chapTargetUsername | Nombre de usuario de destino. Obligatorio si useCHAP=true. Para `ontap-san` y `ontap-san-economy` |
| ONTAP | chapTargetInitiatorSecret | Secreto del iniciador de destino CHAP. Obligatorio si useCHAP=true. Para `ontap-san` y `ontap-san-economy` |

El secreto creado en este paso se va a referenciar en el campo `spec.credentials` del objeto `TridentBackendConfig` que se crea en el siguiente paso.

### Paso 2: crea el `TridentBackendConfig`CR

Ya estás listo para crear tu `TridentBackendConfig` CR. En este ejemplo, se crea un backend que usa el `ontap-san` driver mediante el objeto `TridentBackendConfig` que se muestra a continuación:

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

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

Ahora que creaste la `TridentBackendConfig`CR, puedes verificar el estado. Mira el siguiente ejemplo:

```
kubectl -n trident get tbc backend-tbc-ontap-san
NAME                     BACKEND NAME          BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san    ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success
```

Se creó exitosamente un backend y se vinculó al `TridentBackendConfig` CR.

La fase puede tomar uno de los siguientes valores:

- `Bound`: El `TridentBackendConfig` CR está asociado con un backend, y ese backend contiene `configRef` configurado con el `TridentBackendConfig` uid del CR.
- `Unbound`: Representado mediante `""`. El `TridentBackendConfig` objeto no está vinculado a un backend. Todos los nuevos `TridentBackendConfig` CR están en esta fase por defecto. Después de que la fase cambie, no puede volver a estar sin vincular.
- `Deleting`: El `TridentBackendConfig` CR `deletionPolicy` se configuró para eliminarse. Cuando el `TridentBackendConfig` CR se elimina, pasa al estado de eliminación.
  - Si no existen reclamos de volumen persistentes (PVC) en el backend, eliminar el `TridentBackendConfig` hará que Trident elimine el backend así como el

`TridentBackendConfig` CR.

  - Si hay una o más PVC en el backend, este pasa a un estado de eliminación. El `TridentBackendConfig` CR posteriormente también entra en fase de eliminación. El backend y `TridentBackendConfig` se eliminan solo después de que se eliminen todas las PVC.

- `Lost`: El backend asociado con el `TridentBackendConfig` CR se eliminó accidental o deliberadamente y el `TridentBackendConfig` CR todavía tiene una referencia al backend eliminado. El `TridentBackendConfig` CR todavía puede eliminarse independientemente del `deletionPolicy` valor.
- `Unknown`: Trident no puede determinar el estado ni la existencia del backend asociado con el `TridentBackendConfig` CR. Por ejemplo, si el servidor API no responde o si el `tridentbackends.trident.netapp.io` CRD falta. Esto podría requerir intervención.

En esta etapa, ¡el backend se ha creado correctamente! Hay varias operaciones que también se pueden gestionar, como "actualizaciones y eliminaciones del backend".

### (Opcional) Paso 4: obtén más detalles

Puedes ejecutar el siguiente comando para obtener más información sobre tu backend:

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

```
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success   ontap-san        delete
```

Además, también puedes obtener un volcado YAML/JSON de `TridentBackendConfig`.

```
kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: 2021-04-21T20:45:11Z
  finalizers:
    - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound
```

`backendInfo` contiene el `backendName` y el `backendUUID` del backend que se creó en respuesta al `TridentBackendConfig` CR. El `lastOperationStatus` campo representa el estado de la última operación del `TridentBackendConfig` CR, que puede ser activada por el usuario (por ejemplo, si el usuario cambió algo en `spec`) o activada por Trident (por ejemplo, durante reinicios de Trident). Puede ser Success o Failed. `phase` representa el estado de la relación entre el `TridentBackendConfig` CR y el backend. En el ejemplo anterior, `phase` tiene el valor Bound, lo que significa que el `TridentBackendConfig` CR está asociado con el backend.

Puedes ejecutar el `kubectl -n trident describe tbc <tbc-cr-name>` comando para obtener detalles de los registros de eventos.

| | |
|---|---|
| **ADVERTENCIA** | No puedes actualizar ni eliminar un backend que contenga un objeto asociado `TridentBackendConfig` usando `tridentctl`. Para entender los pasos para cambiar entre `tridentctl` y `TridentBackendConfig`, "ver aquí". |

# 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
```

**NOTA**

- 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
```

**NOTA** 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
```

**NOTA** 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.

# Crea y gestiona clases de almacenamiento

## Crear una clase de almacenamiento

Configura un objeto StorageClass de Kubernetes y crea la clase de almacenamiento para indicarle a Trident cómo aprovisionar volúmenes.

### Configura un objeto StorageClass de Kubernetes

El "Objeto de Kubernetes StorageClass" identifica a Trident como el aprovisionador que se usa para esa clase y le indica a Trident cómo aprovisionar un volumen. Por ejemplo:

```
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
```

Consulta "Objetos de Kubernetes y Trident" para ver detalles sobre cómo las clases de almacenamiento interactúan con el `PersistentVolumeClaim` y los parámetros para controlar cómo Trident aprovisiona volúmenes.

### Crear una clase de almacenamiento

Después de crear el objeto StorageClass, puedes crear la clase de almacenamiento. Ejemplos de storage class proporciona algunos ejemplos básicos que puedes usar o modificar.

**Pasos**

1. Este es un objeto de Kubernetes, así que usa `kubectl` para crearlo en Kubernetes.

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

2. Ahora deberías ver una clase de almacenamiento **basic-csi** tanto en Kubernetes como en Trident, y Trident debería haber descubierto los pools en el backend.

```
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"
        ]
      }
    }
  ]
}
```


#### Ejemplos de storage class

Trident proporciona "definiciones de clases de almacenamiento simples para backends específicos".

Alternativamente, puedes editar `sample-input/storage-class-csi.yaml.templ` archivo que viene con el instalador y reemplazar *`BACKEND_TYPE`* con el nombre del controlador de almacenamiento.

```
./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
```

## Administra clases de almacenamiento

Puedes ver las clases de almacenamiento existentes, establecer una clase de almacenamiento predeterminada, identificar el backend de la clase de almacenamiento y eliminar clases de almacenamiento.

#### Ver las clases de almacenamiento existentes

- Para ver las clases de almacenamiento de Kubernetes existentes, ejecuta el siguiente comando:

```
kubectl get storageclass
```

- Para ver los detalles de la clase de almacenamiento de Kubernetes, ejecuta el siguiente comando:

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

- Para ver las clases de almacenamiento sincronizadas de Trident, ejecuta el siguiente comando:

```
tridentctl get storageclass
```

- Para ver los detalles de la clase de almacenamiento sincronizado de Trident, ejecuta el siguiente comando:

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

#### Establece una clase de almacenamiento predeterminada

Kubernetes 1.6 agregó la capacidad de establecer una clase de almacenamiento predeterminada. Esta es la clase de almacenamiento que se usará para aprovisionar un volumen persistente si un usuario no especifica una en una reclamación de volumen persistente (PVC).

- Define una clase de almacenamiento predeterminada estableciendo la anotación `storageclass.kubernetes.io/is-default-class` como true en la definición de la clase de almacenamiento. Según la especificación, cualquier otro valor o la ausencia de la anotación se interpreta como false.
- Puedes configurar una clase de almacenamiento existente para que sea la clase de almacenamiento predeterminada usando el siguiente comando:

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

- De manera similar, puedes eliminar la anotación de clase de almacenamiento predeterminada usando el siguiente comando:

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

También hay ejemplos en el instalador de Trident que incluyen esta anotación.

| | |
|---|---|
| **NOTA** | Solo debe haber una clase de almacenamiento predeterminada en tu clúster a la vez. Kubernetes no impide técnicamente que tengas más de una, pero se comportará como si no hubiera ninguna clase de almacenamiento predeterminada. |

#### Identifica el backend para una clase de almacenamiento

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

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

#### Eliminar una clase de almacenamiento

Para eliminar una clase de almacenamiento de Kubernetes, ejecuta el siguiente comando:

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

`<storage-class>` debe reemplazarse con tu clase de almacenamiento.

Cualquier volumen persistente que se haya creado a través de esta clase de almacenamiento permanecerá intacto, y Trident seguirá administrándolos.

| | |
|---|---|
| **NOTA** | Trident aplica un espacio en blanco `fsType` para los volúmenes que crea. Para los backends iSCSI, se recomienda aplicarlo `parameters.fsType` en el StorageClass. Deberías eliminar los StorageClasses existentes y volver a crearlos con `parameters.fsType` especificado. |

# Aprovisiona y gestiona volúmenes

## Aprovisiona un volumen

Crea un PersistentVolumeClaim (PVC) que use el StorageClass de Kubernetes configurado para solicitar acceso al PV. Luego puedes montar el PV en un pod.

### Descripción general

Una "*PersistentVolumeClaim*" (PVC) es una solicitud de acceso al PersistentVolume en el clúster.

El PVC se puede configurar para solicitar almacenamiento de un tamaño o modo de acceso determinados. Usando el StorageClass asociado, el administrador del clúster puede controlar más que solo el tamaño y el modo de acceso de PersistentVolume, como el rendimiento o el nivel de servicio.

Después de crear el PVC, puedes montar el volumen en un pod.

### Crea el PVC

**Pasos**

1. Crea el PVC.

```
kubectl create -f pvc.yaml
```

2. Verifica el estado del PVC.

```
kubectl get pvc
```

```
NAME          STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS AGE
pvc-storage   Bound    pv-name    1Gi        RWO                    5m
```

1. Monta el volumen en un pod.

```
kubectl create -f pv-pod.yaml
```

**NOTA** Puedes monitorear el progreso usando `kubectl get pod --watch`.

2. Verifica que el volumen esté montado en `/my/mount/path`.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

3. Ahora puedes eliminar el Pod. La aplicación del Pod ya no existirá, pero el volumen permanecerá.

```
kubectl delete pod pv-pod
```

#### Manifiestos de muestra

### Manifiestos de muestra de PersistentVolumeClaim

Estos ejemplos muestran opciones básicas de configuración de PVC.

#### PVC con acceso RWO

Este ejemplo muestra un PVC básico con acceso RWO que está asociado con una StorageClass llamada `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

#### PVC con NVMe/TCP

Este ejemplo muestra un PVC básico para NVMe/TCP con acceso RWO que está asociado con una StorageClass llamada `protection-gold`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san-nvme
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: protection-gold
```

### Ejemplos de manifiestos de Pod

Estos ejemplos muestran configuraciones básicas para conectar el PVC a un pod.

#### Configuración básica

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: storage
      persistentVolumeClaim:
       claimName: pvc-storage
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: storage
```

#### Configuración básica de NVMe/TCP

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

Consulta "Objetos de Kubernetes y Trident" para ver detalles sobre cómo las clases de almacenamiento interactúan con el `PersistentVolumeClaim` y los parámetros para controlar cómo Trident aprovisiona volúmenes.

## Ampliar volúmenes

Trident permite a los usuarios de Kubernetes ampliar sus volúmenes después de que se crean. Encuentra información sobre las configuraciones necesarias para ampliar volúmenes iSCSI, NFS, SMB, NVMe/TCP y FC.

### Expandir un volumen iSCSI

Puedes expandir un volumen persistente iSCSI (PV) usando el aprovisionador CSI.

| NOTA | La expansión de volumen iSCSI es compatible con los `ontap-san`, `ontap-san-economy`, `solidfire-san` controladores y requiere Kubernetes 1.16 y versiones posteriores. |
|---|---|

#### Paso 1: Configura la StorageClass para admitir la expansión de volumen

Edita la definición de StorageClass para establecer el campo `allowVolumeExpansion` en `true`.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Para un StorageClass ya existente, edítalo para incluir el `allowVolumeExpansion` parámetro.

#### Paso 2: Crea un PVC con el StorageClass que creaste

Edita la definición de PVC y actualiza el `spec.resources.requests.storage` para reflejar el nuevo tamaño deseado, que debe ser mayor que el tamaño original.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident crea un volumen persistente (PV) y lo asocia con este Persistent Volume Claim (PVC).

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

#### Paso 3: define un pod que adjunta el PVC

Conecta el PV a un pod para que pueda ser redimensionado. Hay dos escenarios cuando redimensionas un PV iSCSI:

- Si el PV está conectado a un pod, Trident expande el volumen en el backend de almacenamiento, vuelve a escanear el dispositivo y redimensiona el sistema de archivos.
- Al intentar redimensionar un PV no conectado, Trident expande el volumen en el backend de almacenamiento. Después de que el PVC se vincula a un pod, Trident vuelve a escanear el dispositivo y redimensiona el sistema de archivos. Luego, Kubernetes actualiza el tamaño del PVC después de que la operación de expansión se haya completado correctamente.

En este ejemplo, se crea un pod que usa el `san-pvc`.

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

#### Paso 4: expandir el PV

Para cambiar el tamaño del PV que se ha creado de 1Gi a 2Gi, edita la definición de PVC y actualiza el `spec.resources.requests.storage` a 2Gi.

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

**Paso 5: valida la expansión**

Puedes validar que la expansión funcionó correctamente verificando el tamaño del PVC, PV y el volumen de Trident:

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

### Expandir un volumen FC

Puedes expandir un volumen persistente (PV) FC usando el aprovisionador CSI.

**NOTA** La expansión del volumen FC es compatible con el `ontap-san` driver y requiere Kubernetes 1.16 y versiones posteriores.

#### Paso 1: Configura la StorageClass para admitir la expansión de volumen

Edita la definición de StorageClass para establecer el campo `allowVolumeExpansion` en `true`.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Para un StorageClass ya existente, edítalo para incluir el `allowVolumeExpansion` parámetro.

#### Paso 2: Crea un PVC con el StorageClass que creaste

Edita la definición de PVC y actualiza el `spec.resources.requests.storage` para reflejar el nuevo tamaño deseado, que debe ser mayor que el tamaño original.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident crea un volumen persistente (PV) y lo asocia con este Persistent Volume Claim (PVC).

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

#### Paso 3: define un pod que adjunta el PVC

Conecta el PV a un pod para que pueda ser redimensionado. Hay dos escenarios cuando se redimensiona un PV de FC:

- Si el PV está conectado a un pod, Trident expande el volumen en el backend de almacenamiento, vuelve a escanear el dispositivo y redimensiona el sistema de archivos.
- Al intentar redimensionar un PV no conectado, Trident expande el volumen en el backend de almacenamiento. Después de que el PVC se vincula a un pod, Trident vuelve a escanear el dispositivo y redimensiona el sistema de archivos. Luego, Kubernetes actualiza el tamaño del PVC después de que la

operación de expansión se haya completado correctamente.

En este ejemplo, se crea un pod que usa el `san-pvc`.

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

#### Paso 4: expandir el PV

Para cambiar el tamaño del PV que se ha creado de 1Gi a 2Gi, edita la definición de PVC y actualiza el `spec.resources.requests.storage` a 2Gi.

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

**Paso 5: valida la expansión**

Puedes validar que la expansión funcionó correctamente verificando el tamaño del PVC, PV y el volumen de Trident:

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

### Expandir un volumen NFS

Trident admite la expansión de volumen para PVs NFS aprovisionados en `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup` y `azure-netapp-files` backends.

#### Paso 1: Configura la StorageClass para admitir la expansión de volumen

Para cambiar el tamaño de un PV NFS, el administrador primero debe configurar la clase de almacenamiento para permitir la expansión del volumen configurando el campo `allowVolumeExpansion` en `true`:

```
cat storageclass-ontapnas.yaml
```

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

Si ya creaste una clase de almacenamiento sin esta opción, simplemente puedes editar la clase de

almacenamiento existente usando `kubectl edit storageclass` para permitir la expansión del volumen.

#### Paso 2: Crea un PVC con el StorageClass que creaste

```
cat pvc-ontapnas.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Trident debería crear un NFS PV de 20 MiB para este PVC:

```
kubectl get pvc
NAME           STATUS    VOLUME
CAPACITY     ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

#### Paso 3: expandir el PV

Para cambiar el tamaño del PV de 20 MiB recién creado a 1 GiB, edita el PVC y establece `spec.resources.requests.storage` en 1 GiB:

```
kubectl edit pvc ontapnas20mb
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
# ...
```

**Paso 4: valida la expansión**

Puedes validar que el cambio de tamaño funcionó correctamente verificando el tamaño del PVC, PV y el volumen Trident:

```
kubectl get pvc ontapnas20mb
NAME           STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi
RWO            ontapnas        4m44s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

## Entiende los límites del subsistema RWX NVMe

ReadWriteMany (RWX) volúmenes que usan el protocolo NVMe tienen un límite de escalabilidad de 64 nodos por volumen. A continuación se incluyen las limitaciones, se explica la arquitectura del subsistema NVMe involucrada y se describen los pasos de resolución necesarios.

#### Entiende el límite de 64 nodos

Si planeas usar volúmenes ReadWriteMany (RWX) con el protocolo NVMe, un solo volumen RWX NVMe no puede ser montado por más de 64 nodos en un clúster de Kubernetes.

No programes cargas de trabajo que monten el mismo RWX NVMe PersistentVolumeClaim en más de 64 nodos.

Esta limitación solo se aplica a los volúmenes RWX que usan el protocolo NVMe.

#### Entiende los modelos de subsistemas NVMe

#### Modelo de subsistema por volumen (Trident releases earlier than 26.02)

En las versiones de Trident anteriores a 26.02, los volúmenes RWX NVMe se aprovisionan utilizando un modelo de subsistema por volumen. Cada volumen RWX NVMe se asigna a su propio subsistema NVMe dedicado en ONTAP.

Este modelo es sencillo, pero tiene un límite de escalabilidad inferior. En clústeres de Kubernetes grandes, los límites del controlador de subsistema se alcanzan rápidamente porque cada volumen RWX consume un subsistema dedicado.

#### Modelo de super-subsystem (introducido en Trident 26.02)

A partir de Trident 26.02, los volúmenes RWX NVMe utilizan un modelo de super-subsubsistema compartido. Varios volúmenes RWX NVMe comparten el mismo subsistema NVMe.

Cada super-subsystem admite hasta 1024 namespaces (volúmenes). Este modelo mejora significativamente la escalabilidad para cargas de trabajo RWX y reduce la probabilidad de alcanzar los límites del subsistema ONTAP.

Cada volumen RWX NVMe admite hasta 64 nodos.

### Identifica los síntomas de error

Si creas o adjuntas volúmenes RWX NVMe a escala, podrías ver errores similares a los siguientes:

```
Maximum number of controllers reached. No more controllers can be created.
```

Este error indica que se ha alcanzado el límite del controlador del subsistema ONTAP NVMe.

### Soluciona errores de límite de subsistema

Para superar las limitaciones de los subsistemas por volumen y aprovechar el modelo de supersubsistema, actualiza a Trident 26.02 o posterior.

### Actualiza Trident para aplicar el modelo de super-subsystem

Para aplicar el modelo de super-subsistema para volúmenes RWX NVMe:

1. Actualiza Trident a la versión 26.02 o posterior.
2. Reduce todos los pods que usan volúmenes RWX NVMe a cero réplicas.
3. Verifica que ninguna carga de trabajo esté utilizando activamente volúmenes RWX NVMe.
4. Vuelve a escalar los pods.

Esta secuencia de reinicio garantiza que los volúmenes RWX NVMe se adjunten usando el modelo de super-subsystem.

- Esta limitación solo se aplica a los volúmenes RWX que usan el protocolo NVMe.
- El límite de 64 nodos se aplica por cada volumen RWX NVMe.
- Otros modos de acceso y otros protocolos no están afectados.

# Escalabilidad del controlador

Trident introduce la escalabilidad del controlador mediante una mejor concurrencia entre varios controladores de almacenamiento. Los clientes pueden identificar qué controladores de Trident admiten la escalabilidad del controlador en disponibilidad general y cuáles están disponibles como vista previa técnica en Trident 26.02. Esto permite tomar decisiones informadas de implementación y una gestión de riesgos adecuada para entornos Kubernetes escalables.

## Conceptos clave y definiciones

#### Escalabilidad del controlador

La escalabilidad del controlador se refiere a la capacidad del controlador Trident para procesar múltiples operaciones de almacenamiento en paralelo en lugar de serializarlas detrás de un único bloqueo. Estas operaciones incluyen la creación, eliminación, redimensionamiento de volúmenes, la creación y eliminación de instantáneas, la publicación y despublicación de volúmenes y la gestión de backend.

Cuando la escalabilidad del controlador está activada, las operaciones en diferentes volúmenes y backends se realizan de forma concurrente. Esto aumenta el rendimiento y reduce el tiempo de operación de extremo a extremo en entornos con un gran número de PersistentVolumeClaim y VolumeSnapshot operaciones concurrentes.

## Compatibilidad con la escalabilidad del controlador

Trident admite la escalabilidad del controlador con diferentes niveles de madurez según el controlador de almacenamiento.

#### Disponibilidad general

Los siguientes drivers soportan la escalabilidad del controlador en disponibilidad general en Trident 26.02:

- `ontap-san`
- `ontap-nas`
- `google-cloud-netapp-volumes`

| | |
|---|---|
| **NOTA** | Los controladores `google-cloud-netapp-volumes` y `google-cloud-netapp-volumes-san` son diferentes. Solo `google-cloud-netapp-volumes` es compatible. No uses `google-cloud-netapp-volumes-san` en configuraciones backend ni en ejemplos. |

## Habilita la escalabilidad del controlador

La escalabilidad del controlador se controla mediante la `enableConcurrency` opción de configuración. Esta opción debe habilitarse explícitamente durante la instalación de Trident o actualizando una implementación existente.

#### Despliegue del operador Trident

Para habilitar la escalabilidad del controlador con el operador Trident, establece `enableConcurrency` en `true` en el recurso personalizado `TridentOrchestrator`.

**Nueva instalación**

Crea o edita el `TridentOrchestrator` CR con `enableConcurrency` configurado en `true`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableConcurrency: true
```

Aplica la CR:

```
kubectl apply -f tridentorchestrator_cr.yaml
```

**Instalación existente**

Aplica un parche a la `TridentOrchestrator` CR existente para habilitar la escalabilidad del controlador:

```
kubectl patch torc trident --type=merge -p
'{"spec":{"enableConcurrency":true}}'
```

Verifica que se haya aplicado el ajuste:

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableConcurrency}'
```

#### Despliegue con Helm

Para activar la escalabilidad del controlador con Helm, establece el valor `enableConcurrency` a `true`.

**Nueva instalación**

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace --set enableConcurrency=true
```

**Instalación existente**

```
helm upgrade trident netapp-trident/trident-operator --namespace trident
--set enableConcurrency=true
```

Como alternativa, establece `enableConcurrency` en `true` en un archivo `values.yaml` personalizado:

```
# values.yaml
enableConcurrency: true
```

Luego instala o actualiza usando el archivo de valores:

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace -f values.yaml
```

#### implementación de tridentctl

Para habilitar la escalabilidad del controlador con `tridentctl`, pasa la bandera `--enable-concurrency` durante la instalación.

### Nueva instalación

```
tridentctl install -n trident --enable-concurrency
```

### Instalación existente

Para habilitar la escalabilidad del controlador en una implementación basada en `tridentctl`, desinstala y vuelve a instalar con el flag:

```
tridentctl uninstall -n trident
tridentctl install -n trident --enable-concurrency
```

#### Verifica que la escalabilidad del controlador esté activada

Después de habilitar la escalabilidad del controlador, verifica que el controlador Trident se está ejecutando con la concurrencia habilitada revisando los registros del pod del controlador:

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

Deberías ver una entrada de registro que indica que la concurrencia está activada.

#### Avance técnico

Los siguientes drivers admiten la escalabilidad del controlador como vista previa técnica en Trident 26.02:

- `nas-eco`
- `san-eco`

Para estos drivers:

- La concurrencia de controladores está disponible para evaluación y pruebas
- El comportamiento puede cambiar en futuras versiones
- No se recomienda usarlo en entornos de producción

**Comportamiento de la concurrencia**

Cuando la escalabilidad del controlador está activada:

- Trident reemplaza el bloqueo global único por bloqueos detallados por recurso
- Las operaciones que modifican el mismo recurso se serializan para mantener la coherencia de los datos
- Las operaciones que solo leen de un recurso pueden realizarse simultáneamente con otras operaciones de lectura en ese recurso
- Trident limita las solicitudes simultáneas de la API de ONTAP a 20 por LIF de gestión para evitar sobrecargar los sistemas de almacenamiento backend
- Si varios backends comparten el mismo LIF de gestión, comparten este límite de 20 solicitudes

**Limitaciones y consideraciones conocidas**

Las siguientes consideraciones aplican a la escalabilidad del controlador:

- La concurrencia se gestiona internamente por el controlador Trident
- En esta versión no hay límites de concurrencia configurables por el usuario
- El rendimiento global depende de:
  - El controlador de almacenamiento en uso
  - Capacidad de respuesta del backend
  - Rendimiento del servidor API de Kubernetes
- Una alta concurrencia puede aumentar la carga en los sistemas de almacenamiento backend

**Advertencias y limitaciones**

Las siguientes limitaciones se aplican en Trident 26.02:

- El comportamiento de la escalabilidad del controlador no es idéntico en todos los drivers
- Los controladores de vista previa técnica pueden mostrar:
  - Rendimiento inconsistente bajo carga alta
  - Cambios en el comportamiento entre versiones
- La depuración de operaciones concurrentes puede ser más compleja debido a la ejecución paralela
- Las métricas y los registros pueden mostrar la salida de operaciones intercaladas

**Recomendaciones**

- Usa controladores de disponibilidad general (GA) para entornos de producción que requieran alta escalabilidad
- Evalúa los controladores de technical preview en entornos que no sean de producción
- Supervisa el rendimiento del backend y del controlador cuando trabajas a escala

- Evita asumir el orden de las operaciones en los scripts de automatización

## Ampliación automática de volúmenes

La expansión automática de volúmenes permite que los Volúmenes Persistentes aprovisionados por Trident crezcan automáticamente cuando la capacidad utilizada alcanza un umbral definido. Esta capacidad reduce la sobrecarga operativa y ayuda a evitar la interrupción de la aplicación causada por el agotamiento de la capacidad. La expansión automática de volúmenes se implementa utilizando Autogrow Policies. Una Autogrow Policy define:

- El umbral de utilización que activa la expansión
- La cantidad en que crece el volumen
- El tamaño máximo que puede alcanzar el volumen

| | |
|---|---|
| **NOTA** | Los volúmenes aumentan de tamaño automáticamente cuando se supera el umbral de utilización definido. Los volúmenes nunca se reducen automáticamente. |

### Requisitos

Antes de configurar la expansión automática de volumen, asegúrate de que se cumplen los siguientes requisitos:

- Trident 26.02 o posterior
- Permisos de control de acceso basados en roles para crear `TridentAutogrowPolicy` recursos personalizados
- StorageClasses configurados con `allowVolumeExpansion: true`
- Protocolos ONTAP compatibles:
  - Network File System (NFS)
  - Internet Small Computer Systems Interface (iSCSI)
  - Non-Volatile Memory Express (NVMe)
  - Protocolo de canal de fibra (FCP)

### Limitaciones

- Los volúmenes de bloques sin procesar ONTAP Non-Volatile Memory Express anteriores a ONTAP 9.16.1 no admiten la expansión automática.
- Para volúmenes de red de área de almacenamiento, si `growthAmount` es menor o igual a 50 mebibytes, Trident aumenta automáticamente el valor a 51 mebibytes antes de redimensionar, siempre que el tamaño resultante no supere `maxSize`.
- En entornos brownfield, es posible que la expansión automática no funcione para ciertos volúmenes existentes debido al comportamiento de migración de publicación de volúmenes.
- Cuando un volumen alcanza `maxSize`, no se produce ninguna expansión adicional.
- Protocolos compatibles para la expansión automática del volumen:
  - Network File System (NFS)

- Internet Small Computer Systems Interface (iSCSI)
- Non-Volatile Memory Express (NVMe)
- Protocolo de canal de fibra (FCP)

### Provisiona volúmenes con la política de crecimiento automático

La política de Autogrow puede configurarse a dos niveles:

- Nivel de clase de almacenamiento: establece el valor predeterminado para todos los volúmenes (mediante anotación)
- Nivel de PVC: anula el valor por defecto de la clase de almacenamiento (usando anotación)

### Crear una política de Autogrow

Las políticas de crecimiento automático permiten la expansión automática de volúmenes cuando los volúmenes alcanzan un umbral de capacidad definido.

Asegúrate de tener:

- Trident 26.02 o posterior instalado
- Permisos de control de acceso basados en roles para crear `TridentAutogrowPolicy` recursos
- Comprender los requisitos de crecimiento de la carga de trabajo

Una política de crecimiento automático define cómo los volúmenes se expanden automáticamente cuando alcanzan un umbral de capacidad definido.

Puedes crear políticas Autogrow en cualquier punto de tu flujo de trabajo:

- Antes de que se creen StorageClasses y volúmenes
- Después de que existan StorageClasses
- Después de que se aprovisionan los volúmenes

Esta flexibilidad te permite introducir la expansión automática sin recrear los recursos existentes.

#### Especificaciones de la política Autogrow

Las políticas de crecimiento automático son recursos personalizados de Kubernetes definidos como sigue:

| Campo | Descripción | Formato | Requerido | Ejemplo | Predeterminado |
|---|---|---|---|---|---|
| nombre | Identificador único de política | Cadena | Sí | production-db-policy | Ninguno |
| usedThreshold | Porcentaje de capacidad que activa la expansión | Cadena de porcentaje | Sí | "80%" | Ninguno |
| growthAmount | Cantidad a aumentar cuando se alcanza el umbral | Porcentaje o tamaño | No | "10%" o "5Gi" | "10%" |
| maxSize | Límite máximo de tamaño de volumen | Cantidad de Kubernetes | No | "500Gi" | Ilimitada |

### Crear una política de Autogrow

**Pasos**

1. Crea un archivo YAML que defina tu política de crecimiento automático de volúmenes:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

2. Aplica la política a tu clúster:

```
kubectl apply -f autogrow-policy.yaml
```

3. Verifica que se haya creado la política:

```
kubectl get tridentautogrowpolicy standard-autogrow
```

**Salida esperada**

```
NAME                USED THRESHOLD   GROWTH AMOUNT   STATE
standard-autogrow   80%              10%             Success
```

### Política establece

Después de crear una política, Trident valida la especificación y asigna uno de los siguientes estados:

| Estado | Descripción | Acción requerida |
|---|---|---|
| Éxito | La política está validada y lista para usar. | Ninguno. |
| Con errores | Errores de validación detectados. | Revisa y corrige la especificación. |
| Eliminando | Eliminación en curso. | Espera a que termine. |

### Asocia una política con un StorageClass

Puedes asociar una política de crecimiento automático con un StorageClass usando la anotación `trident.netapp.io/autogrowPolicy`. Todos los volúmenes aprovisionados desde ese StorageClass heredan la política.

**NOTA** El StorageClass debe tener `allowVolumeExpansion: true`.

**Pasos**

1. Crea o modifica un StorageClass con la anotación Autogrow Policy:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

2. Aplica el StorageClass:

```
kubectl apply -f storageclass.yaml
```

3. Verifica la anotación:

```
kubectl get storageclass ontap-gold -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

**Salida esperada**

```
production-db-policy
```

### Precedencia de políticas

Cuando se establecen anotaciones de Autogrow Policy tanto en un StorageClass como en un PVC, Trident aplica las siguientes reglas de precedencia:

1. **La anotación del PVC tiene prioridad.** Si un PVC establece `trident.netapp.io/autogrowPolicy`, ese valor siempre se utiliza.
2. **La anotación StorageClass se aplica solo cuando el PVC no tiene ninguna anotación.**
3. **Si ninguno de los dos tiene la anotación,** no se aplica ninguna Autogrow Policy.

| Anotación de StorageClass | Anotación de PVC | Comportamiento eficaz |
|---|---|---|
| `trident.netapp.io/autogrowPolicy: standard-agp` | No establecido | Usos `standard-agp`. |

| Anotación de StorageClass | Anotación de PVC | Comportamiento eficaz |
|---|---|---|
| `trident.netapp.io/autogrowPolicy: standard-agp` | `trident.netapp.io/autogrowPolicy: logs-policy` | Utiliza `logs-policy` (PVC anula StorageClass). |
| `trident.netapp.io/autogrowPolicy: standard-agp` | `trident.netapp.io/autogrowPolicy: "none"` | No Autogrow Policy (PVC desactiva autogrow). |
| No establecido | `trident.netapp.io/autogrowPolicy: dev-policy` | Usos `dev-policy`. |
| No establecido | No establecido | Sin política de Autogrow. |

### Ejemplos de configuración

Los siguientes ejemplos muestran configuraciones comunes de Autogrow Policy para diferentes casos de uso.

#### Política conservadora para bases de datos de producción

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

#### Almacenamiento de logs con incrementos de crecimiento fijos

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: log-storage-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

#### Política mínima con valores predeterminados

Cuando omites `growthAmount` y `maxSize`, Trident usa los valores predeterminados (`10%` crecimiento, tamaño ilimitado):

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: logs-policy
spec:
  usedThreshold: "85%"
```

**Política con un maxSize personalizado y growthAmount predeterminado**

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: default-ga-policy
spec:
  usedThreshold: "70%"
  maxSize: "100Gi"
```

**Crecimiento agresivo con maxSize ilimitado**

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

**Política con porcentajes fraccionarios**

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: precise-policy
spec:
  usedThreshold: "80.28%"
  growthAmount: "10.65%"
  maxSize: "100Gi"
```

**NOTA** Se admiten porcentajes fraccionarios. Si especificas más de tres decimales, Trident redondea el valor a tres decimales.

#### NAS StorageClass con Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

#### SAN StorageClass con Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

## Gestiona las políticas de Autogrow

Después de crear las políticas de Autogrow, puedes verlas, actualizarlas y eliminarlas cuando lo necesites. También puedes monitorear qué volúmenes están usando una política determinada.

### Ver políticas de Autogrow

#### Lista todas las políticas

Usa `kubectl` para listar todas las Autogrow Policies en tu cluster:

```
kubectl get tridentautogrowpolicy
```

O también, usa `tridentctl`:

```
tridentctl get autogrowpolicy
```

#### Ver detalles de la policy

Para ver la especificación completa y el estado de una política:

```
kubectl describe tridentautogrowpolicy production-db-policy
```

Para ver una política con sus volúmenes asociados en formato YAML:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

### Actualiza una política de Autogrow

Puedes modificar una política existente para cambiar su umbral, cantidad de crecimiento o tamaño máximo. Los cambios surten efecto de inmediato para todos los volúmenes que usan la política.

| | |
|---|---|
| **IMPORTANTE** | Los cambios afectan a todos los volúmenes que actualmente usan la política. Prueba primero los cambios en un entorno de no producción cuando sea posible. |

#### Pasos

1. Edita la política:

   ```
   kubectl edit tridentautogrowpolicy production-db-policy
   ```

2. Modifica los campos `spec` según lo necesites:

   ```
   spec:
     usedThreshold: "75%"     # Changed from 80%
     growthAmount: "20%"      # Changed from 10%
     maxSize: "1Ti"           # Changed from 500Gi
   ```

3. Guarda y sal. Los cambios surten efecto inmediatamente.

#### Consideraciones de actualización

- **Efecto inmediato:** Todos los volúmenes que usan la política adoptan nuevos parámetros en la siguiente evaluación de crecimiento.
- **No es necesario reiniciar el volumen:** Los cambios se aplican a la siguiente operación de crecimiento.
- **Prueba primero:** Valida los cambios en un entorno que no sea de producción cuando sea posible.
- **Comunica los cambios:** avisa a los equipos cuando modifiques las políticas compartidas.

### Eliminar una política de Autogrow

Las políticas de Autogrow usan la protección de finalizador para evitar el borrado accidental mientras los volúmenes las están usando activamente.

**Pasos**

1. Elimina la política:

```
kubectl delete tridentautogrowpolicy production-db-policy
```

2. Si los volúmenes siguen utilizando la política, la eliminación entra en un estado `Deleting`. Revisa qué volúmenes se ven afectados:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

3. Elimina la política de cada volumen afectado. Elige una de las siguientes opciones:

   - **Opción A: desactivar explícitamente autogrow** estableciendo la anotación en `"none"`:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

   - **Opción B: eliminar por completo la anotación:**

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

#### Comportamiento de borrado

| Escenario | Comportamiento |
|---|---|
| Ningún volumen usa la política | La política se elimina inmediatamente. |
| Los volúmenes están usando la política | La política entra `Deleting` en estado. Un finalizador bloquea la finalización hasta que se eliminan todos los volúmenes. |
| Todos los volúmenes se eliminan de la política | Se eliminan los finalizadores y se elimina la política. |

### Supervisa el uso de la política Autogrow

#### Comprobar volúmenes usando una política

```
tridentctl get autogrowpolicy production-db-policy -o json | jq '.volumes'
```

#### Averigua qué política usa un volumen

```
kubectl get pvc database-pvc -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

#### Supervisa los eventos de políticas

```
kubectl get events --field-selector
involvedObject.kind=TridentAutogrowPolicy
```

### Protocolos compatibles

Autogrow es compatible con los siguientes protocolos de almacenamiento:

- NFS
- iSCSI
- FCP
- NVMe

| | |
|---|---|
| **NOTA** | Para volúmenes SAN, si el `growthAmount` configurado es de 50 MiB o menos, Trident aumenta automáticamente la cantidad de crecimiento a 51 MB para la operación de redimensionamiento, siempre que el tamaño resultante no supere `maxSize`. |

### Limitaciones conocidas

- **Volúmenes de bloques sin procesar ONTAP NVMe:** Los volúmenes creados con versiones de ONTAP anteriores a 9.16.1 no admiten autogrow.
- **Volúmenes existentes (implantaciones brownfield):** Es posible que Autogrow no funcione para los volúmenes existentes aunque se aplique una Autogrow Policy válida. Esto se debe a una migración en curso de publicaciones de volúmenes. Para confirmar que la migración ha terminado, revisa los registros del controlador Trident para mensajes de `"Migration completed"`.

### Preguntas frecuentes

#### ¿Cuándo evalúa Trident el umbral?

Trident supervisa continuamente el uso del volumen. Cuando la capacidad utilizada supera el `usedThreshold`, Trident crea una solicitud interna de redimensionamiento y amplía el volumen por la `growthAmount` configurada.

Por ejemplo, esta política activa la expansión al 80% de la capacidad y aumenta el volumen un 10% cada vez, hasta un máximo de 500 GiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

#### ¿Puedo aplicar una política después de que los volúmenes ya estén aprovisionados?

Sí. Puedes crear una Autogrow Policy en cualquier momento y aplicarla a los PVC existentes añadiendo o actualizando la anotación `trident.netapp.io/autogrowPolicy`. No necesitas volver a crear el PVC ni el StorageClass.

Para aplicar una política a un PVC existente:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Para aplicar una política a un StorageClass existente:

```
kubectl annotate storageclass ontap-gold \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

#### ¿Qué pasa si configuro una Autogrow Policy tanto en el StorageClass como en el PVC?

La anotación de PVC siempre tiene prioridad. Si un PVC tiene la `trident.netapp.io/autogrowPolicy` anotación, Trident usa ese valor sin importar lo que especifique StorageClass. Consulta "Precedencia de políticas" para más detalles.

Por ejemplo, dada esta StorageClass:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

Y este PVC que anula la política de StorageClass:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
  storageClassName: ontap-gold
```

Trident utiliza `logs-policy` para `database-pvc`, no `standard-agp`.

#### ¿Cómo desactivo el autogrow para un volumen específico?

Establece la anotación de PVC en `"none"`. Esto anula cualquier política de nivel StorageClass para ese volumen:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

Puedes comprobar que autogrow está desactivado:

```
kubectl get pvc <pvc-name> -o jsonpath
='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

**Salida esperada**

```
none
```

#### ¿Qué pasa cuando un volumen llega a maxSize?

Trident deja de expandir el volumen. No se crean más solicitudes de redimensionamiento para ese volumen, incluso si el uso sigue aumentando más allá del `usedThreshold`.

Por ejemplo, con esta política, Trident deja de aumentar el volumen una vez que llega a los 100 GiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: capped-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Para permitir un crecimiento ilimitado, omite `maxSize` o configúralo en `0`:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: unlimited-policy
spec:
  usedThreshold: "85%"
  growthAmount: "10%"
```

#### ¿Puedo cambiar una política sin reiniciar los volúmenes?

Sí. Cuando actualizas una política, todos los volúmenes que usan esa política adoptan los nuevos parámetros en la siguiente evaluación de crecimiento. No se requieren reinicios de volúmenes.

Para actualizar una política en vigor:

```
kubectl edit tridentautogrowpolicy production-db-policy
```

Modifica los campos según lo necesites:

```
spec:
  usedThreshold: "75%"    # Changed from 80%
  growthAmount: "20%"     # Changed from 10%
  maxSize: "1Ti"          # Changed from 500Gi
```

Guarda y sal. Verifica la política actualizada:

```
kubectl get tridentautogrowpolicy production-db-policy
```

**Salida esperada**

```
NAME                    USED THRESHOLD   GROWTH AMOUNT   STATE
production-db-policy    75%              20%             Success
```

#### ¿Por qué mi policy está en estado Failed?

Un estado `Failed` indica que la especificación de la política contiene errores de validación. Ejecuta el siguiente comando para ver los detalles del error:

```
kubectl describe tridentautogrowpolicy <policy-name>
```

Las causas comunes incluyen un `usedThreshold` inválido (debe ser 1–99%), un `growthAmount` que excede `maxSize` o un formato de cantidad de Kubernetes inválido. Corrige la especificación y vuelve a aplicar:

```
kubectl apply -f autogrow-policy.yaml
```

#### ¿Por qué no puedo eliminar una policy?

Las políticas utilizan la protección del finalizador. Si los volúmenes todavía están utilizando la política, la eliminación entra en un `Deleting` estado y espera hasta que todos los volúmenes se eliminen de la política.

Identifica los volúmenes afectados:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Luego elimina la anotación de cada PVC:

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Después de que se eliminan todos los volúmenes, se libera el finalizador y se elimina la política.

#### ¿Funciona autogrow con todos los backends de ONTAP?

Autogrow es compatible con NFS, iSCSI, FCP y los protocolos NVMe. Sin embargo, los volúmenes de bloques sin procesar NVMe requieren ONTAP 9.16.1 o una versión posterior.

Es posible que los volúmenes existentes en las implementaciones brownfield necesiten que la migración de publicación de volúmenes se complete antes de que el autogrow surta efecto. Verifica el estado de la migración revisando los registros del controlador Trident:

```
kubectl logs -l app=trident-controller -n trident | grep "Migration
completed"
```

Los siguientes ejemplos de StorageClass muestran autogrow configurado para backends NAS y SAN:

**Backend NAS**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

**Backend SAN**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

#### ¿Cuál es la cantidad mínima de crecimiento para los volúmenes SAN?

Para volúmenes SAN, la cantidad mínima de crecimiento efectivo es de 51 MB. Si configuras un `growthAmount` de 50 MiB o menos, Trident aumenta automáticamente el crecimiento a 51 MB para la operación de redimensionamiento.

Por ejemplo, esta política establece un `growthAmount` de `"40Mi"`, pero Trident aplica un crecimiento de 51 MB para cualquier volumen SAN que lo use:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-minimal-policy
spec:
  usedThreshold: "85%"
  growthAmount: "40Mi"
  maxSize: "100Gi"
```

Para evitar este ajuste automático, establece `growthAmount` en un valor mayor a 50 MiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-policy
spec:
  usedThreshold: "85%"
  growthAmount: "100Mi"
  maxSize: "500Gi"
```

## Importar volúmenes

Puedes importar volúmenes de almacenamiento existentes como un PV de Kubernetes usando `tridentctl import` o creando una Persistent Volume Claim (PVC) con anotaciones de importación de Trident.

### Descripción general y consideraciones

Puedes importar un volumen en Trident para:

- Containerizar una aplicación y reutilizar su conjunto de datos existente
- Usa un clon de un conjunto de datos para una aplicación efímera
- Reconstruye un clúster de Kubernetes fallido
- Migra datos de aplicaciones durante la recuperación ante desastres

#### Consideraciones

Antes de importar un volumen, revisa las siguientes consideraciones.

- Trident solo puede importar volúmenes ONTAP de tipo RW (lectura-escritura). Los volúmenes de tipo DP (protección de datos) son volúmenes destino de SnapMirror. Debes romper la relación de espejo antes de importar el volumen en Trident.
- Sugerimos importar volúmenes sin conexiones activas. Para importar un volumen en uso activo, clona el volumen y luego haz la importación.

| | |
|---|---|
| **ADVERTENCIA** | Esto es especialmente importante para los volúmenes de bloques, ya que Kubernetes no sabría de la conexión anterior y podría conectar fácilmente un volumen activo a un pod. Esto puede provocar la corrupción de datos. |

- Aunque `StorageClass` debe especificarse en una PVC, Trident no usa este parámetro durante la importación. Las clases de almacenamiento se usan durante la creación del volumen para seleccionar entre los pools disponibles según las características del almacenamiento. Como el volumen ya existe, no se requiere seleccionar un pool durante la importación. Así que la importación no fallará incluso si el volumen existe en un backend o pool que no coincide con la clase de almacenamiento especificada en la PVC.
- El tamaño del volumen existente se determina y se configura en el PVC. Después de que el controlador de almacenamiento importa el volumen, el PV se crea con un ClaimRef al PVC.
  - La política de recuperación se establece inicialmente en `retain` el PV. Después de que Kubernetes vincula correctamente el PVC y el PV, la política de recuperación se actualiza para que coincida con la política de recuperación de la Storage Class.
  - Si la política de recuperación de la Storage Class es `delete`, el volumen de almacenamiento se eliminará cuando se elimine el PV.
- De forma predeterminada, Trident administra el PVC y renombra el volumen FlexVol y el LUN en el backend. Puedes pasar la `--no-manage` bandera para importar un volumen no administrado y la `--no-rename` bandera para conservar el nombre del volumen.
  - `--no-manage`* - Si usas la `--no-manage`flag, Trident no realiza ninguna operación adicional en el PVC ni en el PV durante el ciclo de vida de los objetos. El volumen de almacenamiento no se elimina cuando se elimina el PV y otras operaciones como clonar el volumen y redimensionar el volumen también se ignoran.
  - `--no-rename`* - Si usas la `--no-rename`flag, Trident conserva el nombre del volumen existente al importar volúmenes y administra el ciclo de vida de los volúmenes. Esta opción solo es compatible con los `ontap-nas`, `ontap-san` (incluidos los sistemas ASA r2), y `ontap-san-economy`drivers.

| | |
|---|---|
| **CONSEJO** | Estas opciones son útiles si quieres usar Kubernetes para cargas de trabajo en contenedores, pero si prefieres, puedes gestionar el ciclo de vida del volumen de almacenamiento fuera de Kubernetes. |

- Se añade una anotación al PVC y al PV que cumple una doble función: indicar que el volumen fue importado y si el PVC y el PV están gestionados. Esta anotación no debe modificarse ni eliminarse.

**Importar un volumen**

Puedes importar un volumen usando `tridentctl import` o creando un PVC con anotaciones de importación de Trident.

| | |
|---|---|
| **NOTA** | Si usas anotaciones de PVC, no necesitas descargar ni usar `tridentctl` para importar el volumen. |

**Usando tridentctl**

**Pasos**

1. Crea un archivo PVC (por ejemplo, `pvc.yaml`) que se usará para crear el PVC. El archivo PVC debe incluir `name`, `namespace`, `accessModes` y `storageClassName`. Opcionalmente, puedes especificar `unixPermissions` en tu definición de PVC.

   El siguiente es un ejemplo de una especificación mínima:

   ```
   kind: PersistentVolumeClaim
   apiVersion: v1
   metadata:
     name: my_claim
     namespace: my_namespace
   spec:
     accessModes:
       - ReadWriteOnce
     storageClassName: my_storage_class
   ```

   **NOTA** Incluye solo los parámetros necesarios. Parámetros adicionales como el nombre de PV o el tamaño del volumen pueden hacer que falle el comando de import.

2. Usa el `tridentctl import` comando para especificar el nombre del backend de Trident que contiene el volumen y el nombre que identifica de forma única el volumen en el almacenamiento (por ejemplo: ONTAP FlexVol, Element Volume). El argumento `-f` es necesario para especificar la ruta al archivo PVC.

   ```
   tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-file>
   ```

**Uso de anotaciones de PVC**

**Pasos**

1. Crea un archivo PVC YAML (por ejemplo, `pvc.yaml`) con las anotaciones de importación de Trident necesarias. El archivo PVC debe incluir:

   - `name` y `namespace` en metadatos
   - `accessModes`, `resources.requests.storage` y `storageClassName` en spec
   - Anotaciones:
     - `trident.netapp.io/importOriginalName`: nombre del volumen en el backend
     - `trident.netapp.io/importBackendUUID`: UUID del backend donde existe el volumen
     - `trident.netapp.io/notManaged` (*Opcional*): configúralo en `"true"` para volúmenes no gestionados. El valor predeterminado es `"false"`.

       El siguiente es un ejemplo de especificación para importar un volumen gestionado:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. Aplica el archivo YAML de PVC a tu clúster de Kubernetes:

```
kubectl apply -f <pvc-file>.yaml
```

Trident importará automáticamente el volumen y lo vinculará al PVC.

### Ejemplos

Revisa los siguientes ejemplos de importación de volúmenes para los controladores compatibles.

#### ONTAP NAS y ONTAP NAS FlexGroup

Trident admite la importación de volúmenes usando los `ontap-nas` y `ontap-nas-flexgroup` controladores.

**NOTA**

- Trident no admite la importación de volúmenes usando el `ontap-nas-economy` driver.
- Los `ontap-nas` y `ontap-nas-flexgroup` controladores no permiten nombres de volúmenes duplicados.

Cada volumen creado con el `ontap-nas` controlador es un volumen FlexVol en el clúster ONTAP. Importar volúmenes FlexVol con el `ontap-nas` controlador funciona igual. Un volumen FlexVol que ya existe en un clúster ONTAP se puede importar como un `ontap-nas` PVC. De manera similar, los volúmenes FlexGroup se pueden importar como `ontap-nas-flexgroup` PVCs.

#### Ejemplos de ONTAP NAS usando tridentctl

Los siguientes ejemplos muestran cómo importar volúmenes gestionados y no gestionados usando `tridentctl`.

**Volumen administrado**

El siguiente ejemplo importa un volumen denominado `managed_volume` en un backend denominado `ontap_nas`:

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

**Volumen no administrado**

Al usar el `--no-manage` argumento, Trident no cambia el nombre del volumen.

El siguiente ejemplo importa `unmanaged_volume` en el `ontap_nas` backend:

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-
file> --no-manage

+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

**Ejemplos de ONTAP NAS usando anotaciones de PVC**

Los siguientes ejemplos muestran cómo importar volúmenes gestionados y no gestionados usando anotaciones de PVC.

**Volumen administrado**

El siguiente ejemplo importa un volumen de 1GiB `ontap-nas` llamado `ontap_volume1` desde backend `81abcb27-ea63-49bb-b606-0a5315ac5f21` con el modo de acceso RWO establecido usando anotaciones PVC:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

**Volumen no administrado**

El siguiente ejemplo importa 1Gi `ontap-nas` volumen llamado `ontap-volume2` desde backend `34abcb27-ea63-49bb-b606-0a5315ac5f34` con el modo de acceso RWO configurado usando anotaciones PVC:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <umanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-
0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

#### ONTAP SAN

Trident admite la importación de volúmenes usando los `ontap-san` (iSCSI, NVMe/TCP y FC) y `ontap-san-economy` drivers.

Trident puede importar volúmenes SAN de ONTAP FlexVol que contienen un solo LUN. Esto es coherente con el `ontap-san` driver, que crea un volumen FlexVol para cada PVC y un LUN dentro del volumen FlexVol. Trident importa el volumen FlexVol y lo asocia con la definición de PVC. Trident puede importar volúmenes `ontap-san-economy` que contienen varios LUN.

Los siguientes ejemplos muestran cómo importar volúmenes gestionados y no gestionados:

**Volumen administrado**

Para los volúmenes administrados, Trident cambia el nombre del FlexVol volumen al `pvc-<uuid>` formato y el LUN dentro del FlexVol volumen a `lun0`.

El siguiente ejemplo importa el `ontap-san-managed` volumen FlexVol que está presente en el `ontap_san_default` backend:

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

**Volumen no administrado**

El siguiente ejemplo importa `unmanaged_example_volume` en el `ontap_san` backend:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Si tienes LUN asignados a igroups que comparten un IQN con un IQN de nodo Kubernetes, como se muestra en el siguiente ejemplo, recibirás el error: `LUN already mapped to initiator(s) in this group`. Necesitarás eliminar el iniciador o desasignar el LUN para importar el volumen.

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

### Ejemplos de ONTAP SAN-economy

Los siguientes ejemplos muestran cómo importar volúmenes gestionados y no gestionados para el `ontap-san-economy` backend.

**Volumen administrado**

Al importar un volumen gestionado, Trident toma posesión del FlexVol y le cambia el nombre. Debes tener en cuenta este cambio de nombre cuando importes varios LUN del mismo FlexVol.

El siguiente ejemplo importa `lun1` desde el FlexVol `toimport` como un volumen gestionado llamado `vol-managed-saneco`:

```
tridentctl import volume vol-managed-saneco toimport/lun1 -f
import1.yaml
```

Después de importar `lun1`, Trident cambia el nombre del FlexVol (por ejemplo, a `trident_lun_pool_xyz`). Para importar LUN adicionales del mismo FlexVol, usa el nuevo nombre de FlexVol:

```
tridentctl import volume vol-managed-saneco trident_lun_pool_xyz/lun2
-f import2.yaml
```

**NOTA** El backend `ontap-san-economy` importa un LUN a la vez. Puedes automatizar varias importaciones usando un script.

**Volumen no administrado**

Al importar un volumen no gestionado, Trident no se apropia del FlexVol. Sin embargo, el FlexVol y el LUN deben seguir las convenciones de nomenclatura de Trident.

**Formato de nomenclatura de FlexVol**

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- `STORAGEPREFIX` es el valor de `storagePrefix` en tu configuración de backend. El valor predeterminado es `trident`.
- `RANDOMSTRING` es cualquier cadena que elijas.

**Requisito de nomenclatura LUN**

El LUN debe llamarse `lun0`.

**Ejemplo**

Si tu `storagePrefix` es `xyz`, la ruta completa al LUN es:

```
trident_lun_pool_xyz_randomstring/lun0
```

#### Elemento

Trident admite el software NetApp Element y la importación de volúmenes NetApp HCI usando el controlador

`solidfire-san`.

| | |
|---|---|
| **NOTA** | El controlador Element admite nombres de volumen duplicados. Sin embargo, Trident devuelve un error si hay nombres de volumen duplicados. Como solución alternativa, clona el volumen, ponle un nombre de volumen único e importa el volumen clonado. |

El siguiente ejemplo importa un `element-managed` volumen en el backend `element_default`.

```
tridentctl import volume element_default element-managed -f pvc-basic-
import.yaml -n trident -d

+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

#### Azure NetApp Files

Trident admite la importación de volúmenes usando el `azure-netapp-files` driver.

| | |
|---|---|
| **NOTA** | Para importar un volumen de Azure NetApp Files, identifica el volumen por su ruta de volumen. La ruta de volumen es la parte de la ruta de exportación del volumen después de `:/`. Por ejemplo, si la ruta de montaje es `10.0.0.2:/importvol1`, la ruta de volumen es `importvol1`. |

El siguiente ejemplo importa un `azure-netapp-files` volumen en el backend `azurenetappfiles_40517` con la ruta de volumen `importvol1`.

```
tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage   |
file     | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true    |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

#### Google Cloud NetApp Volumes

Trident admite la importación de volúmenes usando el `google-cloud-netapp-volumes` driver.

El siguiente ejemplo importa un volumen en backend `backend-tbc-gcnv1` con el volumen `testvoleasiaeast1`.

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

El siguiente ejemplo importa un `google-cloud-netapp-volumes` volumen cuando hay dos volúmenes presentes en la misma región:

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

## Personaliza los nombres y las etiquetas de los volúmenes

Con Trident, puedes asignar nombres y etiquetas significativos a los volúmenes que creas. Esto te ayuda a identificar y asignar fácilmente los volúmenes a sus respectivos recursos de Kubernetes (PVCs). También puedes definir plantillas en el backend para crear nombres y etiquetas de volumen personalizados; cualquier volumen que crees, importes o clones se ajustará a las plantillas.

### Antes de empezar

Soporte para nombres y etiquetas de volumen personalizables:

- Operaciones de crear, importar y clonar volúmenes.
- En el caso del `ontap-nas-economy`driver, solo el nombre del volumen Qtree cumple con la plantilla de nombre.
- En el caso del `ontap-san-economy`driver, solo el nombre LUN cumple con la plantilla de nombre.

### Limitaciones

- Los nombres de volúmenes personalizados solo son compatibles con los drivers on-premises de ONTAP.
- Las etiquetas personalizadas solo son compatibles con los `ontap-san`, `ontap-nas` y `ontap-nas-flexgroup` controladores.
- Los nombres de volúmenes personalizados no se aplican a los volúmenes existentes.

### Comportamientos clave de los nombres de volúmenes personalizables

- Si se produce un error debido a una sintaxis no válida en una plantilla de nombre, la creación del backend

falla. Sin embargo, si la aplicación de la plantilla falla, el volumen se nombrará según la convención de nomenclatura existente.

- El prefijo de almacenamiento no es aplicable cuando un volumen se nombra usando una plantilla de nombres de la configuración del backend. Cualquier valor de prefijo que quieras se puede agregar directamente a la plantilla.

### Ejemplos de configuración de backend con plantilla de nombre y etiquetas

Se pueden definir plantillas de nombres personalizadas en el nivel raíz y/o en el nivel de pool.

#### Ejemplo de nivel raíz

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.Requ
estName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

**Ejemplo a nivel de pool**

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels": {
        "labelname": "label1",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels": {
        "cluster": "label2",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

**Ejemplos de plantillas de nombres**

**Ejemplo 1**:

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

**Ejemplo 2**:

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

**Puntos a considerar**

1. En el caso de las importaciones de volumen, las etiquetas se actualizan solo si el volumen existente tiene etiquetas en un formato específico. Por ejemplo: `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. En el caso de importaciones de volúmenes administrados, el nombre del volumen sigue la plantilla de nombre definida en el nivel raíz en la definición del backend.
3. Trident no admite el uso de un operador de corte con el prefijo de almacenamiento.
4. Si las plantillas no dan como resultado nombres de volumen únicos, Trident agregará algunos caracteres aleatorios para crear nombres de volumen únicos.
5. Si el nombre personalizado de un volumen NAS economy supera los 64 caracteres, Trident nombrará los volúmenes según la convención de nomenclatura existente. Para todos los demás controladores ONTAP, si el nombre del volumen supera el límite de caracteres, el proceso de creación del volumen falla.

## Comparte un volumen NFS entre espacios de nombres

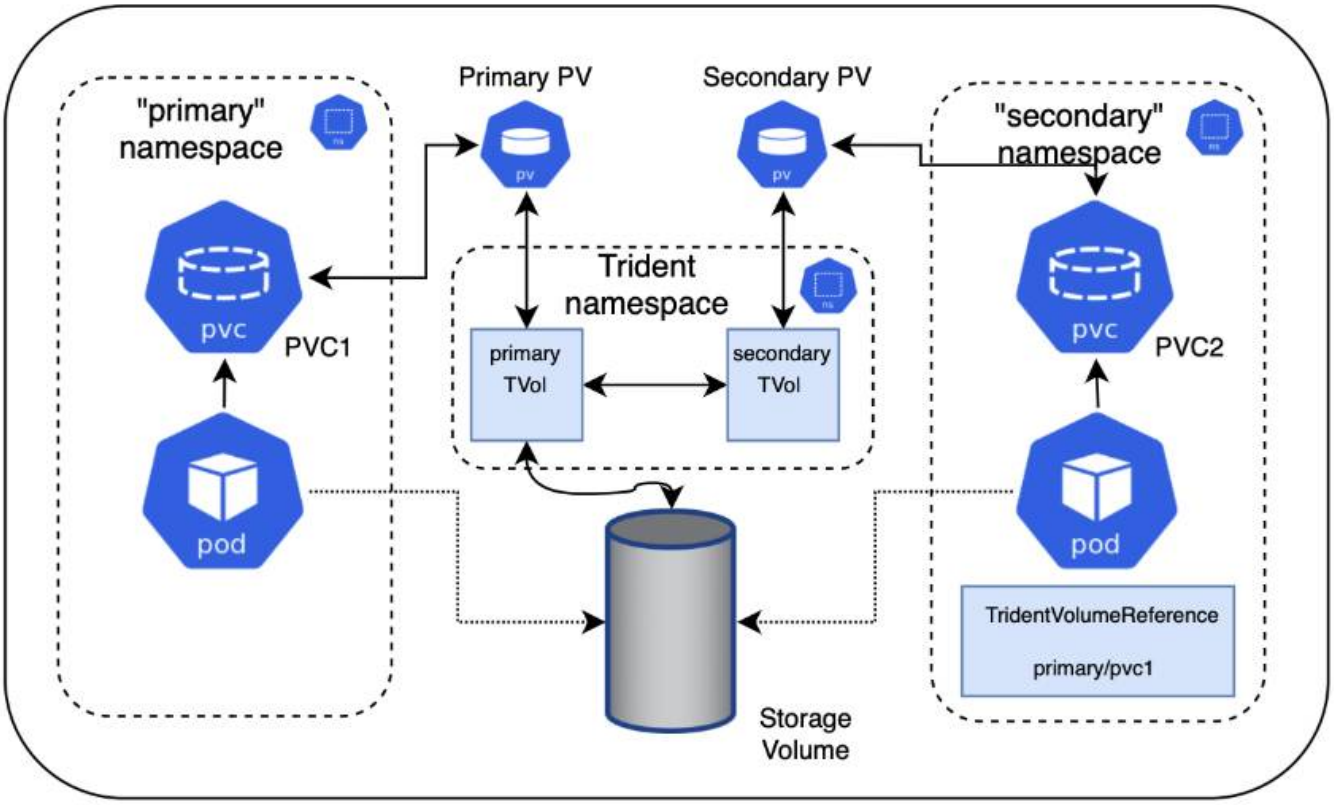
Con Trident, puedes crear un volumen en un espacio de nombres primario y compartirlo en uno o más espacios de nombres secundarios.

**Funciones**

El CR TridentVolumeReference te permite compartir de forma segura volúmenes NFS ReadWriteMany (RWX) entre uno o varios espacios de nombres de Kubernetes. Esta solución nativa de Kubernetes tiene los siguientes beneficios:

- Múltiples niveles de control de acceso para garantizar la seguridad
- Funciona con todos los controladores de volumen NFS de Trident
- No depende de tridentctl ni de ninguna otra función no nativa de Kubernetes

Este diagrama ilustra el uso compartido de volúmenes NFS en dos espacios de nombres de Kubernetes.




**Inicio rápido**

Puedes configurar el uso compartido de volúmenes NFS en unos pocos pasos.

1 **Configura el PVC de origen para compartir el volumen**

El propietario del espacio de nombres de origen otorga permiso para acceder a los datos en el PVC de origen.

2 **Otorga permiso para crear una CR en el espacio de nombres de destino**

El administrador del clúster otorga permiso al propietario del espacio de nombres de destino para crear el CR TridentVolumeReference.

3 **Crea TridentVolumeReference en el espacio de nombres de destino**

El propietario del espacio de nombres de destino crea el CR TridentVolumeReference para hacer referencia al PVC de origen.

4 **Crea la PVC subordinada en el espacio de nombres de destino**

El propietario del espacio de nombres de destino crea el PVC subordinado para usar la fuente de datos del PVC de origen.

### Configura los espacios de nombres de origen y destino

Para garantizar la seguridad, el uso compartido entre espacios de nombres requiere la colaboración y acción del propietario del espacio de nombres de origen, el administrador del clúster y el propietario del espacio de nombres de destino. El rol del usuario se designa en cada paso.

**Pasos**

1. **Propietario del espacio de nombres de origen:** Crea el PVC (`pvc1`) en el espacio de nombres de origen que da permiso para compartir con el espacio de nombres de destino (`namespace2`) usando la anotación `shareToNamespace`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

   Trident crea el PV y su volumen de almacenamiento NFS de backend.

**NOTA**

- Puedes compartir el PVC con varios espacios de nombres usando una lista separada por comas. Por ejemplo, `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4`.
- Puedes compartir a todos los espacios de nombres usando `*`. Por ejemplo, `trident.netapp.io/shareToNamespace: *`
- Puedes actualizar el PVC para incluir la anotación `shareToNamespace` en cualquier momento.

2. **Administrador del clúster:** Asegúrate de que existe el RBAC adecuado para conceder permiso al propietario del espacio de nombres de destino para crear el CR TridentVolumeReference en el espacio de nombres de destino.
3. **Propietario del espacio de nombres de destino:** crea una TridentVolumeReference CR en el espacio de nombres de destino que hace referencia al espacio de nombres de origen `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Propietario del espacio de nombres de destino:** crea un PVC (`pvc2`) en el espacio de nombres de destino (`namespace2`) usando la anotación `shareFromPVC` para designar el PVC de origen.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

**NOTA** El tamaño del PVC de destino debe ser menor o igual que el PVC de origen.

#### Resultados

Trident lee la `shareFromPVC` anotación en el PVC de destino y crea el PV de destino como un volumen subordinado sin recurso de almacenamiento propio que apunta al PV de origen y comparte el recurso de almacenamiento del PV de origen. El PVC y el PV de destino aparecen vinculados de forma normal.

### Eliminar un volumen compartido

Puedes eliminar un volumen que está compartido entre varios espacios de nombres. Trident quitará el acceso al volumen en el espacio de nombres de origen y mantendrá el acceso para los demás espacios de nombres que comparten el volumen. Cuando se eliminan todos los espacios de nombres que hacen referencia al volumen, Trident elimina el volumen.

### Usa `tridentctl get` para consultar volúmenes subordinados

Usando la utilidad [`tridentctl`, puedes ejecutar el comando `get` para obtener los volúmenes subordinados. Para más información, consulta el enlace:../trident-reference/trident-cl.html[`tridentctl` comandos y

opciones].

```
Usage:
  tridentctl get [option]
```

Banderas:

- `-h, --help`: ayuda para volúmenes.
- `--parentOfSubordinate string`: limitar la consulta al volumen fuente subordinado.
- `--subordinateOf string`: limitar la consulta a los subordinados del volumen.

#### Limitaciones

- Trident no puede evitar que los espacios de nombres de destino escriban en el volumen compartido. Deberías usar el bloqueo de archivos u otros procesos para evitar sobrescribir los datos del volumen compartido.
- No puedes revocar el acceso al PVC de origen eliminando las anotaciones `shareToNamespace` o `shareFromNamespace` o eliminando el `TridentVolumeReference` CR. Para revocar el acceso, tienes que borrar el PVC subordinado.
- Las instantáneas, clones y la replicación no son posibles en volúmenes subordinados.

#### Para más información

Para obtener más información sobre el acceso a volúmenes entre espacios de nombres:

- Mira la demo en "NetAppTV".

## Clonar volúmenes en distintos espacios de nombres

Con Trident, puedes crear nuevos volúmenes usando volúmenes existentes o instantáneas de volúmenes de un espacio de nombres diferente dentro del mismo clúster de Kubernetes.

#### Prerrequisitos

Antes de clonar volúmenes, asegúrate de que los backends de origen y destino sean del mismo tipo y tengan la misma clase de almacenamiento.

| | |
|---|---|
| **NOTA** | La clonación entre espacios de nombres solo se admite para los controladores de almacenamiento `ontap-san` y `ontap-nas`. No se admiten clones de solo lectura. |

#### Inicio rápido

Puedes configurar la clonación de volumen en solo unos pocos pasos.

1. **Configura el PVC de origen para clonar el volumen**

El propietario del espacio de nombres de origen otorga permiso para acceder a los datos en el PVC de origen.

### 2 Otorga permiso para crear una CR en el espacio de nombres de destino

El administrador del clúster otorga permiso al propietario del espacio de nombres de destino para crear el CR TridentVolumeReference.

### 3 Crea TridentVolumeReference en el espacio de nombres de destino

El propietario del espacio de nombres de destino crea el CR TridentVolumeReference para hacer referencia al PVC de origen.

### 4 Crea el PVC clonado en el espacio de nombres de destino

El propietario del espacio de nombres de destino crea PVC para clonar la PVC del espacio de nombres de origen.

### Configura los espacios de nombres de origen y destino

Para garantizar la seguridad, clonar volúmenes entre espacios de nombres requiere la colaboración y acción del propietario del espacio de nombres de origen, el administrador del clúster y el propietario del espacio de nombres de destino. El rol del usuario se designa en cada paso.

**Pasos**

1. **Propietario del espacio de nombres de origen:** Crea el PVC (`pvc1` en el espacio de nombres de origen (`namespace1` que otorga permiso para compartir con el espacio de nombres de destino (`namespace2` usando la anotación `cloneToNamespace`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/cloneToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident crea el PV y su volumen de almacenamiento backend.

**NOTA**

- Puedes compartir el PVC con varios espacios de nombres usando una lista separada por comas. Por ejemplo, `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4`.
- Puedes compartir a todos los espacios de nombres usando `*`. Por ejemplo, `trident.netapp.io/cloneToNamespace: *`
- Puedes actualizar el PVC para incluir la anotación `cloneToNamespace` en cualquier momento.

2. **Administrador del clúster:** Asegúrate de que exista un RBAC adecuado para otorgar permiso al propietario del espacio de nombres de destino para crear la CR TridentVolumeReference en el espacio de nombres de destino (`namespace2`).
3. **Propietario del espacio de nombres de destino:** crea una TridentVolumeReference CR en el espacio de nombres de destino que hace referencia al espacio de nombres de origen `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Propietario del espacio de nombres de destino:** crea un PVC (`pvc2`) en el espacio de nombres de destino (`namespace2`) usando la `cloneFromPVC` o `cloneFromSnapshot`, y `cloneFromNamespace` anotaciones para designar el PVC de origen.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

#### Limitaciones

- Para los PVC aprovisionados usando controladores ontap-nas-economy, no se admiten clones de solo lectura.

## Replica volúmenes usando SnapMirror

Trident admite relaciones de espejo entre un volumen de origen en un clúster y el volumen de destino en el clúster emparejado para replicar datos para recuperación ante desastres. Puedes usar una Custom Resource Definition (CRD) con espacio de nombres, llamada Trident Mirror Relationship (TMR), para realizar las siguientes operaciones:

- Crear relaciones de espejo entre volúmenes (PVCs)
- Eliminar relaciones de espejo entre volúmenes
- Rompe las relaciones espejo
- Promociona el volumen secundario durante condiciones de desastre (conmutaciones por error)
- Realiza una transición sin pérdidas de aplicaciones de un clúster a otro (durante conmutaciones por error o migraciones planificadas)

#### Requisitos previos de replicación

Asegúrate de que se cumplan los siguientes requisitos previos antes de comenzar:

##### Clústeres ONTAP

- **Trident**: Trident versión 22.10 o posterior debe existir tanto en los clústeres de Kubernetes de origen como en los de destino que utilizan ONTAP como backend.
- **Licencias**: Las licencias asíncronas de ONTAP SnapMirror usando el paquete Data Protection deben estar habilitadas tanto en el clúster ONTAP de origen como en el clúster de destino. Consulta "Resumen de licencias de SnapMirror en ONTAP" para más información.

  A partir de ONTAP 9.10.1, todas las licencias se entregan como un archivo de licencia NetApp (NLF), que es un único archivo que habilita varias funciones. Consulta "Licencias incluidas con ONTAP One" para más información.

  | | |
  |---|---|
  | **NOTA** | Solo se admite la protección asíncrona de SnapMirror. |

##### Emparejamiento

- **Clúster y SVM**: Los backends de almacenamiento de ONTAP deben estar interconectados. Consulta "Descripción general de clúster y SVM peering" para más información.

  | | |
  |---|---|
  | **IMPORTANTE** | Asegúrate de que los nombres de SVM usados en la relación de replicación entre dos clústeres ONTAP sean únicos. |

- **Trident y SVM**: las SVM remotas emparejadas deben estar disponibles para Trident en el clúster de destino.

##### Controladores compatibles

NetApp Trident admite la replicación de volúmenes con la tecnología NetApp SnapMirror usando clases de

almacenamiento respaldadas por los siguientes controladores: **`ontap-nas`: NFS** `ontap-san`: iSCSI **`ontap-san`: FC** `ontap-san`: NVMe/TCP (requiere como mínimo la versión ONTAP 9.15.1)

| NOTA | La replicación de volúmenes usando SnapMirror no es compatible con sistemas ASA r2. Para información sobre los sistemas ASA r2, consulta "Conoce los sistemas de almacenamiento ASA r2". |
|---|---|

### Crear un PVC reflejado

Sigue estos pasos y usa los ejemplos de CRD para crear una relación de espejo entre los volúmenes primario y secundario.

#### Pasos

1. Realiza los siguientes pasos en el clúster principal de Kubernetes:
   a. Crea un objeto StorageClass con el parámetro `trident.netapp.io/replication: true`.

      **Ejemplo**

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        backendType: "ontap-nas"
        fsType: "nfs"
        trident.netapp.io/replication: "true"
      ```

   b. Crea un PVC con una StorageClass creada previamente.

      **Ejemplo**

      ```
      kind: PersistentVolumeClaim
      apiVersion: v1
      metadata:
        name: csi-nas
      spec:
        accessModes:
        - ReadWriteMany
        resources:
          requests:
            storage: 1Gi
        storageClassName: csi-nas
      ```

   c. Crea un CR de MirrorRelationship con información local.

**Ejemplo**

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Trident obtiene la información interna del volumen y el estado actual de protección de datos (DP) del volumen, luego rellena el campo de estado de MirrorRelationship.

d. Obtén el TridentMirrorRelationship CR para obtener el nombre interno y SVM del PVC.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. Realiza los siguientes pasos en el clúster secundario de Kubernetes:

   a. Crea un StorageClass con el parámetro trident.netapp.io/replication: true.

**Ejemplo**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/replication: true
```

b. Crea un CR MirrorRelationship con información sobre el destino y el origen.

**Ejemplo**

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Trident creará una relación SnapMirror con el nombre de política de relación configurado (o el predeterminado para ONTAP) y la inicializará.

c. Crea un PVC con el StorageClass creado previamente para que actúe como secundario (SnapMirror destino).

**Ejemplo**

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Trident comprobará si existe el TridentMirrorRelationship CRD y fallará al crear el volumen si la relación no existe. Si la relación existe, Trident se asegurará de que el nuevo volumen FlexVol se coloque en una SVM que esté peered con la SVM remota definida en el MirrorRelationship.

### Estados de replicación de volúmenes

Una Trident Mirror Relationship (TMR) es un CRD que representa un extremo de una relación de replicación entre PVCs. La TMR de destino tiene un estado, que le dice a Trident cuál es el estado deseado. La TMR de destino tiene los siguientes estados:

- **Establecido**: el PVC local es el volumen de destino de una relación de espejo, y esta es una nueva relación.
- **Promocionado**: el PVC local es ReadWrite y montable, sin relación de espejo actualmente en vigor.
- **Reestablecido**: el PVC local es el volumen de destino de una relación de espejo y también estaba previamente en esa relación de espejo.
  - El estado restablecido debe usarse si el volumen de destino estuvo alguna vez en una relación con el volumen de origen porque sobrescribe el contenido del volumen de destino.
  - El estado restablecido fallará si el volumen no estaba previamente en una relación con el origen.

### Promociona el PVC secundario durante una conmutación por error no planificada

Realiza el siguiente paso en el clúster secundario de Kubernetes:

- Actualiza el campo *spec.state* de TridentMirrorRelationship a `promoted`.

### Promociona el PVC secundario durante una conmutación por error planificada

Durante una conmutación por error planificada (migración), realiza los siguientes pasos para promover el PVC secundario:

**Pasos**

1. En el clúster primario de Kubernetes, crea una instantánea del PVC y espera hasta que se cree la instantánea.
2. En el clúster primario de Kubernetes, crea el CR SnapshotInfo para obtener detalles internos.

   **Ejemplo**

   ```
   kind: SnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshot-name: csi-nas-snapshot
   ```

3. En el clúster de Kubernetes secundario, actualiza el campo *spec.state* del CR *TridentMirrorRelationship* a *promoted* y *spec.promotedSnapshotHandle* al internalName de la instantánea.
4. En el clúster secundario de Kubernetes, confirma el estado (campo status.state) de TridentMirrorRelationship a promoted.

### Restaura una relación de réplica tras una conmutación por error

Antes de restablecer una relación de espejo, elige el lado que quieres convertir en el nuevo primario.

#### Pasos

1. En el clúster secundario de Kubernetes, asegúrate de que los valores del campo *spec.remoteVolumeHandle* en el TridentMirrorRelationship estén actualizados.
2. En el clúster de Kubernetes secundario, actualiza el campo *spec.mirror* de TridentMirrorRelationship a `reestablished`.

### Operaciones adicionales

Trident admite las siguientes operaciones en los volúmenes primario y secundario:

#### Replica el PVC primario en un nuevo PVC secundario

Asegúrate de que ya tienes un PVC primario y un PVC secundario.

#### Pasos

1. Elimina los PersistentVolumeClaim y TridentMirrorRelationship CRD del clúster secundario (de destino) establecido.
2. Elimina el CRD TridentMirrorRelationship del clúster principal (origen).
3. Crea un nuevo CRD TridentMirrorRelationship en el clúster principal (de origen) para el nuevo PVC secundario (de destino) que quieres establecer.

#### Redimensiona un PVC reflejado, primario o secundario

El PVC se puede redimensionar de forma normal, ONTAP expandirá automáticamente cualquier destino flevxols si la cantidad de datos excede el tamaño actual.

#### Eliminar la replicación de un PVC

Para eliminar la replicación, realiza una de las siguientes operaciones en el volumen secundario actual:

- Elimina el MirrorRelationship en el PVC secundario. Esto rompe la relación de replicación.
- O, actualiza el campo spec.state a *promoted*.

#### Elimina un PVC (que antes estaba replicado)

Trident comprueba si hay PVC replicados y libera la relación de replicación antes de intentar eliminar el volumen.

#### Eliminar un TMR

Eliminar una TMR en un lado de una relación de réplica hace que la TMR restante pase al estado *promoted* antes de que Trident complete la eliminación. Si la TMR seleccionada para eliminar ya está en estado *promoted*, no existe una relación de réplica y la TMR se eliminará y Trident promoverá el PVC local a *ReadWrite*. Esta eliminación libera los metadatos de SnapMirror para el volumen local en ONTAP. Si este volumen se usa en una relación de réplica en el futuro, debe usar una nueva TMR con un estado de replicación de volumen *established* al crear la nueva relación de réplica.

#### Actualizar las relaciones de réplica cuando ONTAP está en línea

Las relaciones de espejo se pueden actualizar en cualquier momento después de que se establecen. Puedes usar los campos `state: promoted` o `state: reestablished` para actualizar las relaciones. Al promover un volumen de destino a un volumen normal ReadWrite, puedes usar *promotedSnapshotHandle* para especificar una instantánea específica a la que restaurar el volumen actual.

#### Actualiza las relaciones de réplica cuando ONTAP está fuera de línea

Puedes usar un CRD para realizar una actualización de SnapMirror sin que Trident tenga conectividad directa con el clúster ONTAP. Consulta el siguiente formato de ejemplo de TridentActionMirrorUpdate:

**Ejemplo**

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` refleja el estado del TridentActionMirrorUpdate CRD. Puede tomar un valor de *Succeeded*, *In Progress* o *Failed*.

## Usa la topología CSI

Trident puede crear y adjuntar volúmenes de forma selectiva a los nodos presentes en un clúster de Kubernetes mediante el uso de "Función de topología CSI".

#### Descripción general

Con la función CSI Topology, el acceso a los volúmenes se puede limitar a un subconjunto de nodos, según las regiones y zonas de disponibilidad. Hoy en día, los proveedores de nube permiten a los administradores de Kubernetes crear nodos que están basados en zonas. Los nodos pueden estar ubicados en diferentes zonas de disponibilidad dentro de una región o en varias regiones. Para facilitar el aprovisionamiento de volúmenes para cargas de trabajo en una arquitectura multizona, Trident utiliza CSI Topology.

**CONSEJO** Conoce más sobre la función CSI Topology "aquí".

Kubernetes ofrece dos modos de enlace de volumen únicos:

- Con `VolumeBindingMode` configurado en `Immediate`, Trident crea el volumen sin ninguna conciencia de topología. La vinculación de volúmenes y el aprovisionamiento dinámico se gestionan cuando se crea la PVC. Esta es la opción predeterminada `VolumeBindingMode` y es adecuada para clústeres que no aplican restricciones de topología. Los volúmenes persistentes se crean sin tener ninguna dependencia de los requisitos de programación del pod solicitante.
- Con `VolumeBindingMode` establecido en `WaitForFirstConsumer`, la creación y vinculación de un volumen persistente para una PVC se retrasa hasta que se programa y crea un pod que usa la PVC. De esta forma, se crean volúmenes para cumplir con las restricciones de programación impuestas por los requisitos de topología.

| NOTA | El `WaitForFirstConsumer`modo de enlace no requiere etiquetas de topología. Esto puede usarse independientemente de la función de topología CSI. |
|---|---|

**Lo que necesitas**

Para utilizar la topología CSI, necesitas lo siguiente:

- Un clúster de Kubernetes ejecutando un "versión de Kubernetes compatible"

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- Los nodos del clúster deben tener etiquetas que permitan reconocer la topología (`topology.kubernetes.io/region` y `topology.kubernetes.io/zone`). Estas etiquetas **deberían estar presentes en los nodos del clúster** antes de instalar Trident para que Trident reconozca la topología.

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

### Paso 1: crea un backend que tenga en cuenta la topología

Los backends de almacenamiento Trident pueden diseñarse para aprovisionar volúmenes selectivamente según las zonas de disponibilidad. Cada backend puede incluir un `supportedTopologies` bloque opcional que representa una lista de zonas y regiones compatibles. Para StorageClasses que hacen uso de dicho backend, solo se creará un volumen si lo solicita una aplicación programada en una región o zona compatible.

Aquí tienes un ejemplo de definición de backend:

**YAML**

```yaml
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-a
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-b
```

**JSON**

```json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-a"
    },
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-b"
    }
  ]
}
```

**NOTA**

`supportedTopologies` se utiliza para proporcionar una lista de regiones y zonas por backend. Estas regiones y zonas representan la lista de valores permitidos que se pueden proporcionar en un StorageClass. Para StorageClasses que contienen un subconjunto de las regiones y zonas proporcionadas en un backend, Trident crea un volumen en el backend.

Puedes definir `supportedTopologies` por grupo de almacenamiento también. Mira el siguiente ejemplo:

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
storage:
  - labels:
      workload: production
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-b
```

En este ejemplo, las `region` y `zone` etiquetas representan la ubicación del grupo de almacenamiento. `topology.kubernetes.io/region` y `topology.kubernetes.io/zone` indican desde dónde se pueden consumir los grupos de almacenamiento.

#### Paso 2: Define StorageClasses que son conscientes de la topología

Según las etiquetas de topología que se proporcionan a los nodos en el clúster, se pueden definir StorageClasses para que contengan información de topología. Esto determinará los grupos de almacenamiento que sirven como candidatos para las solicitudes de PVC realizadas y el subconjunto de nodos que pueden usar los volúmenes proporcionados por Trident.

Mira el siguiente ejemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions:
    - key: topology.kubernetes.io/zone
      values:
        - us-east1-a
        - us-east1-b
    - key: topology.kubernetes.io/region
      values:
        - us-east1
parameters:
  fsType: ext4
```

En la definición de StorageClass proporcionada arriba, `volumeBindingMode` se establece en `WaitForFirstConsumer`. Las PVC que se soliciten con este StorageClass no se procesarán hasta que se haga referencia a ellas en un pod. Y, `allowedTopologies` proporciona las zonas y la región que se van a usar. El `netapp-san-us-east1` StorageClass crea PVC en el backend definido arriba `san-backend-us-east1`.

#### Paso 3: crea y utiliza un PVC

Con el StorageClass creado y asignado a un backend, ahora puedes crear PVCs.

Mira el ejemplo `spec` abajo:

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata: null
name: pvc-san
spec: null
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

Crear un PVC usando este manifiesto daría como resultado lo siguiente:

```
kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

Para que Trident cree un volumen y lo vincule al PVC, usa el PVC en un pod. Mira el siguiente ejemplo:

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

Este podSpec le indica a Kubernetes que programe el pod en los nodos que están presentes en la `us-east1` región y que elija entre cualquier nodo que esté presente en las `us-east1-a` o `us-east1-b` zonas.

Mira el siguiente resultado:

```
kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

#### Actualizar los backends para incluir `supportedTopologies`

Los backends preexistentes se pueden actualizar para incluir una lista de `supportedTopologies` usando `tridentctl backend update`. Esto no afectará a los volúmenes que ya se hayan aprovisionado y solo se usará para PVC posteriores.

#### Encuentra más información

- "Administra recursos para contenedores"
- "nodeSelector"
- "Afinidad y antiafinidad"
- "Taints y tolerations"

## Trabaja con instantáneas

Las instantáneas de volúmenes de Kubernetes de Persistent Volumes (PVs) permiten copias en un momento específico de los volúmenes. Puedes crear una instantánea de un volumen creado usando Trident, importar una instantánea creada fuera de Trident, crear un nuevo volumen a partir de una instantánea existente y recuperar datos de volúmenes desde instantáneas.

#### Descripción general

La instantánea de volumen es compatible con `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files` y `google-cloud-netapp-volumes` controladores.

#### Antes de empezar

Debes tener un controlador de instantáneas externo y definiciones de recursos personalizadas (CRDs) para trabajar con instantáneas. Esto es responsabilidad del orquestador de Kubernetes (por ejemplo: Kubeadm, GKE, OpenShift).

Si tu distribución de Kubernetes no incluye el controlador de instantáneas y los CRD, consulta Implementa un controlador de instantáneas de volumen.

| NOTA | No crees un controlador de instantáneas si vas a crear instantáneas de volumen bajo demanda en un entorno GKE. GKE usa un controlador de instantáneas integrado y oculto. |
|---|---|

### Crear una instantánea de volumen

#### Pasos

1. Crea un `VolumeSnapshotClass`. Para más información, consulta "VolumeSnapshotClass".
   - El `driver` apunta al controlador Trident CSI.
   - `deletionPolicy` puede ser `Delete` o `Retain`. Cuando se establece en `Retain`, la instantánea física subyacente en el clúster de almacenamiento se conserva incluso cuando el objeto `VolumeSnapshot` se elimina.

     **Ejemplo**

     ```
     cat snap-sc.yaml
     ```

     ```
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshotClass
     metadata:
       name: csi-snapclass
     driver: csi.trident.netapp.io
     deletionPolicy: Delete
     ```

2. Crea una instantánea de un PVC existente.

   **Ejemplos**

   - Este ejemplo crea una instantánea de un PVC existente.

     ```
     cat snap.yaml
     ```

     ```
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshot
     metadata:
       name: pvc1-snap
     spec:
       volumeSnapshotClassName: csi-snapclass
       source:
         persistentVolumeClaimName: pvc1
     ```

   - Este ejemplo crea un objeto de instantánea de volumen para un PVC llamado `pvc1` y el nombre de la instantánea se establece en `pvc1-snap`. Un VolumeSnapshot es análogo a un PVC y está asociado con un `VolumeSnapshotContent` objeto que representa la instantánea real.

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                    AGE
pvc1-snap               50s
```

- Puedes identificar el objeto `VolumeSnapshotContent` para el `pvc1-snap` VolumeSnapshot describiéndolo. El `Snapshot Content Name` identifica el objeto VolumeSnapshotContent que sirve a esta instantánea. El parámetro `Ready To Use` indica que la instantánea puede usarse para crear un nuevo PVC.

```
kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
...
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
...
```

### Crear una PVC a partir de una instantánea de volumen

Puedes usar `dataSource` para crear un PVC usando un VolumeSnapshot llamado `<pvc-name>` como fuente de los datos. Después de crear el PVC, puedes adjuntarlo a un pod y usarlo como cualquier otro PVC.

| | |
|---|---|
| **ADVERTENCIA** | El PVC se creará en el mismo backend que el volumen de origen. Consulta "KB: no se puede crear un PVC a partir de una instantánea de PVC de Trident en un backend alternativo". |

El siguiente ejemplo crea el PVC usando `pvc1-snap` como fuente de datos.

```
cat pvc-from-snap.yaml
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

### Importar una instantánea de volumen

Trident admite la "Proceso de instantáneas preaprovisionadas de Kubernetes" para que el administrador del clúster pueda crear un `VolumeSnapshotContent` objeto e importar instantáneas creadas fuera de Trident.

#### Antes de empezar

Trident debe haber creado o importado el volumen primario de la instantánea.

#### Pasos

1. **Administrador del clúster:** Crea un `VolumeSnapshotContent` objeto que hace referencia a la instantánea del backend. Esto inicia el flujo de trabajo de la instantánea en Trident.
   - Especifica el nombre de la instantánea del backend en `annotations` como `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
   - Especifica `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` en `snapshotHandle`. Esta es la única información proporcionada a Trident por el snapshotter externo en la llamada `ListSnapshots`.

     **NOTA** El `<volumeSnapshotContentName>` no siempre puede coincidir con el nombre de la instantánea de backend debido a las restricciones de nomenclatura de CR.

     **Ejemplo**

     El siguiente ejemplo crea un `VolumeSnapshotContent` objeto que hace referencia a la instantánea del backend `snap-01`.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

2. **Cluster admin:** Crea el `VolumeSnapshot` CR que hace referencia al `VolumeSnapshotContent` objeto. Esto solicita acceso para usar el `VolumeSnapshot` en un espacio de nombres determinado.

   **Ejemplo**

   El siguiente ejemplo crea un `VolumeSnapshot` CR llamado `import-snap` que hace referencia al `VolumeSnapshotContent` llamado `import-snap-content`.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content
```

3. **Procesamiento interno (no requiere acción):** el snapshotter externo reconoce el recién creado `VolumeSnapshotContent` y ejecuta la llamada `ListSnapshots`. Trident crea el `TridentSnapshot`.
   - El external snapshotter establece el `VolumeSnapshotContent` en `readyToUse` y el `VolumeSnapshot` en `true`.
   - Trident devuelve `readyToUse=true`.
4. **Cualquier usuario:** Crea un `PersistentVolumeClaim` para hacer referencia al nuevo `VolumeSnapshot`, donde el `spec.dataSource` (o `spec.dataSourceRef`) nombre es el nombre de `VolumeSnapshot`.

**Ejemplo**

El siguiente ejemplo crea una PVC que hace referencia al `VolumeSnapshot` llamado `import-snap`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

### Recupera datos de volumen usando instantáneas

El directorio de instantáneas está oculto de forma predeterminada para facilitar la máxima compatibilidad de los volúmenes aprovisionados usando los `ontap-nas` y `ontap-nas-economy` controladores. Habilita el directorio `.snapshot` para recuperar datos directamente de las instantáneas.

Usa la CLI de ONTAP para restaurar una instantánea de volumen y devolver un volumen a un estado registrado en una instantánea anterior.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

| | |
|---|---|
| **NOTA** | Cuando restauras una copia instantánea, la configuración existente del volumen se sobrescribe. Los cambios realizados en los datos del volumen después de que se creó la copia instantánea se pierden. |

### Restauración de volumen en el lugar desde una instantánea

Trident permite la restauración rápida e in situ de volúmenes a partir de una instantánea mediante la `TridentActionSnapshotRestore` (TASR) CR. Esta CR funciona como una acción imperativa de Kubernetes y no persiste después de que la operación se completa.

Trident admite la restauración de instantáneas en los `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes` y `solidfire-san` controladores.

**Antes de empezar**

Debes tener un PVC enlazado y una instantánea de volumen disponible.

- Verifica que el estado del PVC esté enlazado.

```
kubectl get pvc
```

- Verifica que la instantánea de volumen esté lista para usar.

```
kubectl get vs
```

**Pasos**

1. Crea la CR de TASR. Este ejemplo crea una CR para PVC `pvc1` y volumen snapshot `pvc1-snapshot`.

   **NOTA** El TASR CR debe estar en un espacio de nombres donde existan el PVC y el VS.

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. Aplica la CR para restaurar desde la instantánea. Este ejemplo restaura desde la instantánea `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

**Resultados**

Trident restaura los datos de la instantánea. Puedes verificar el estado de la restauración de la instantánea:

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

**NOTA**

- En la mayoría de los casos, Trident no reintentará automáticamente la operación en caso de fallo. Tendrás que realizar la operación de nuevo.
- Es posible que los usuarios de Kubernetes sin acceso de administrador tengan que recibir permiso del administrador para crear un TASR CR en el espacio de nombres de su aplicación.

### Eliminar un PV con instantáneas asociadas

Al eliminar un volumen persistente con instantáneas asociadas, el volumen Trident correspondiente se actualiza a un "estado Eliminando". Elimina las instantáneas del volumen para eliminar el volumen Trident.

### Implementa un controlador de instantáneas de volumen

Si tu distribución de Kubernetes no incluye el snapshot controller y los CRDs, puedes implementarlos así.

**Pasos**

1. Crea CRD de instantáneas de volumen.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Crea el controlador de instantáneas.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

**NOTA** Si es necesario, abre `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` y actualiza `namespace` a tu espacio de nombres.

**Enlaces relacionados**

- "Instantáneas de volumen"
- "VolumeSnapshotClass"

## Trabaja con instantáneas de grupo de volúmenes

Instantáneas de grupos de volúmenes de Kubernetes de volúmenes persistentes (PVs) NetApp Trident te permite crear instantáneas de varios volúmenes (un grupo de instantáneas de volúmenes). Esta instantánea de grupo de volúmenes representa copias de varios volúmenes tomadas en el mismo momento específico.

**NOTA** VolumeGroupSnapshot es una función beta en Kubernetes con API beta. Kubernetes 1.32 es la versión mínima requerida para VolumeGroupSnapshot.

### Crear instantáneas de grupo de volúmenes

La instantánea del grupo de volúmenes es compatible con los siguientes controladores de almacenamiento:

- `ontap-san`controlador: solo para los protocolos iSCSI y FC, no para el protocolo NVMe/TCP.
- `ontap-san-economy` - solo para el protocolo iSCSI.
- `ontap-nas`

| NOTA | La instantánea del grupo de volúmenes no es compatible con NetApp ASA r2 o los sistemas de almacenamiento AFX. |
|---|---|

#### Antes de empezar

- Asegúrate de que tu versión de Kubernetes sea K8s 1.32 o superior.
- Debes tener un controlador de instantáneas externo y definiciones de recursos personalizadas (CRDs) para trabajar con instantáneas. Esto es responsabilidad del orquestador de Kubernetes (por ejemplo: Kubeadm, GKE, OpenShift).

  Si tu distribución de Kubernetes no incluye el controlador de instantáneas externo y los CRDs, consulta Implementa un controlador de instantáneas de volumen.

  | NOTA | No crees un controlador de instantáneas si vas a crear instantáneas de grupo de volúmenes bajo demanda en un entorno de GKE. GKE usa un controlador de instantáneas integrado y oculto. |
  |---|---|

- En el YAML del controlador de instantáneas, configura la `CSIVolumeGroupSnapshot` feature gate en 'true' para asegurarte de que la instantánea del grupo de volúmenes esté habilitada.
- Crea las clases de instantáneas de grupo de volúmenes necesarias antes de crear una instantánea de grupo de volúmenes.
- Asegúrate de que todos los PVC/volúmenes estén en el mismo SVM para poder crear VolumeGroupSnapshot.

#### Pasos

- Crea un VolumeGroupSnapshotClass antes de crear un VolumeGroupSnapshot. Para obtener más información, consulta "VolumeGroupSnapshotClass".

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- Crea PVC con las etiquetas requeridas usando clases de almacenamiento existentes o agrega estas etiquetas a los PVC existentes.

  El siguiente ejemplo crea la PVC usando `pvc1-group-snap` como origen de datos y la etiqueta

`consistentGroupSnapshot: groupA`. Define la clave y el valor de la etiqueta según tus necesidades.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- Crea un VolumeGroupSnapshot con la misma etiqueta (`consistentGroupSnapshot: groupA`) especificada en el PVC.

  Este ejemplo crea una instantánea de grupo de volúmenes:

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

**Recupera datos de volumen usando una instantánea de grupo**

Puedes restaurar volúmenes persistentes individuales usando las instantáneas individuales que se han creado como parte de la instantánea del grupo de volúmenes. No puedes recuperar la instantánea del grupo de volúmenes como una unidad.

Usa la CLI de ONTAP para restaurar una instantánea de volumen y devolver un volumen a un estado registrado en una instantánea anterior.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

| NOTA | Cuando restauras una copia instantánea, la configuración existente del volumen se sobrescribe. Los cambios realizados en los datos del volumen después de que se creó la copia instantánea se pierden. |
|---|---|

### Restauración de volumen en el lugar desde una instantánea

Trident permite la restauración rápida e in situ de volúmenes a partir de una instantánea mediante la `TridentActionSnapshotRestore` (TASR) CR. Esta CR funciona como una acción imperativa de Kubernetes y no persiste después de que la operación se completa.

Para más información, consulta "Restauración de volumen en el lugar desde una instantánea".

### Eliminar un PV con instantáneas de grupo asociadas

Al eliminar una instantánea de grupo de volúmenes:

- Puedes eliminar VolumeGroupSnapshots como un todo, no instantáneas individuales en el grupo.
- Si se eliminan PersistentVolumes mientras existe una instantánea para ese PersistentVolume, Trident moverá ese volumen a un estado de "eliminación" porque la instantánea debe eliminarse antes de que el volumen pueda eliminarse de manera segura.
- Si se ha creado un clon utilizando una instantánea agrupada y luego se desea eliminar el grupo, comenzará una operación de división en clon y el grupo no se puede eliminar hasta que se complete la división.

### Implementa un controlador de instantáneas de volumen

Si tu distribución de Kubernetes no incluye el snapshot controller y los CRDs, puedes implementarlos así.

**Pasos**

1. Crea CRD de instantáneas de volumen.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. Crea el controlador de instantáneas.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```

**NOTA** Si es necesario, abre `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` y actualiza `namespace` a tu espacio de nombres.

### Enlaces relacionados

- "VolumeGroupSnapshotClass"
- "Instantáneas de volumen"