



# Aprovisione y gestione volúmenes

Trident

NetApp  
September 26, 2025

# Tabla de contenidos

Aprovisione y gestione volúmenes	1
Aprovisione un volumen	1
Descripción general	1
Cree el VP y la RVP	4
Expanda los volúmenes	5
Expanda un volumen iSCSI	5
Expanda un volumen NFS	10
Importar volúmenes	13
Descripción general y consideraciones	13
Importe un volumen	14
Ejemplos	15
Personalizar nombres y etiquetas de volúmenes	20
Antes de empezar	20
Limitaciones	20
Comportamientos clave de los nombres de volúmenes personalizables	20
Ejemplos de configuración de backend con plantilla de nombre y etiquetas	20
Ejemplos de plantillas de nombres	22
Puntos que considerar	23
Comparta un volumen NFS en espacios de nombres	23
Funciones	23
Inicio rápido	24
Configurar los espacios de nombres de origen y destino	25
Elimine un volumen compartido	26
Se utiliza <code>tridentctl get</code> para consultar los volúmenes subordinados	26
Limitaciones	27
Si quiere más información	27
Replicar volúmenes mediante SnapMirror	27
Requisitos previos de replicación	28
Cree una RVP reflejada	28
Estados de replicación de volúmenes	31
Promocione la RVP secundaria durante una conmutación al respaldo no planificada	31
Promocione la RVP secundaria durante una conmutación al respaldo planificada	32
Restaurar una relación de mirroring después de una conmutación al nodo de respaldo	32
Operaciones adicionales	32
Actualice las relaciones de reflejo cuando el ONTAP esté en línea	33
Actualice las relaciones de reflejo cuando la ONTAP esté sin conexión	33
Utilice Topología CSI	34
Descripción general	34
Paso 1: Cree un backend con detección de topología	35
Paso 2: Defina las clases de almacenamiento que tienen en cuenta la topología	37
Paso 3: Cree y utilice un PVC	38
Actualice los back-ends que se van a incluir <code>supportedTopologies</code>	41

Obtenga más información .....	41
Trabajar con instantáneas .....	41
Descripción general .....	41
Cree una copia de Snapshot de volumen .....	42
Cree una RVP a partir de una snapshot de volumen.....	43
Importe una copia de Snapshot de volumen .....	44
Recuperar datos de volumen mediante copias Snapshot .....	46
Restauración de volumen sin movimiento a partir de una copia de Snapshot .....	46
Eliminar un VP con snapshots asociadas .....	48
Implemente una controladora Snapshot de volumen .....	48
Enlaces relacionados .....	49

# Aprovisione y gestione volúmenes

## Aprovisione un volumen

Cree un volumen persistente (VP) y una reclamación de volumen persistente (RVP) que utilice el tipo de almacenamiento de Kubernetes configurado para solicitar acceso al VP. A continuación, puede montar el VP en un pod.

### Descripción general

Un "*Volumen persistente*" (VP) es un recurso de almacenamiento físico provisionado por el administrador del clúster en un clúster de Kubernetes. "*Claim de volumen persistente*" (RVP) es una solicitud para acceder al volumen persistente en el clúster.

La RVP se puede configurar para solicitar almacenamiento de un determinado tamaño o modo de acceso. Mediante el StorageClass asociado, el administrador del clúster puede controlar mucho más que el tamaño de los volúmenes persistentes y el modo de acceso, como el rendimiento o el nivel de servicio.

Después de crear el VP y la RVP, puede montar el volumen en un pod.

### Manifiestos de muestra

#### Manifiesto de muestra de volumen persistente

Este manifiesto de ejemplo muestra un PV básico de 10Gi que está asociado con StorageClass `basic-csi`.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-storage
  labels:
    type: local
spec:
  storageClassName: basic-csi
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/my/host/path"
```

## Manifiestos de muestra de PersistentVolumeClaim

Estos ejemplos muestran opciones básicas de configuración de PVC.

### PVC con acceso RWO

Este ejemplo muestra una PVC básica con acceso RWO que está asociada con una clase de almacenamiento 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

En este ejemplo se muestra una PVC básica para NVMe/TCP con acceso RWO asociada con una clase de almacenamiento llamada `protection-gold`.

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san-nvme
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 300Mi
  storageClassName: protection-gold
```

## Muestras de manifiesto de POD

Estos ejemplos muestran configuraciones básicas para conectar la RVP 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
```

## Configuración de NVMe/TCP básica

```
---
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    resources: {}
    volumeMounts:
    - mountPath: "/usr/share/nginx/html"
      name: task-pv-storage
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  volumes:
  - name: task-pv-storage
    persistentVolumeClaim:
      claimName: pvc-san-nvme
```

## Cree el VP y la RVP

### Pasos

1. Cree el VP.

```
kubectl create -f pv.yaml
```

2. Compruebe el estado de PV.

```
kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM
STORAGECLASS  REASON    AGE
pv-storage    4Gi       RWO           Retain          Available
7s
```

3. Cree la RVP.

```
kubectl create -f pvc.yaml
```

#### 4. Verifique el estado de la RVP.

```
kubectl get pvc
NAME          STATUS  VOLUME          CAPACITY  ACCESS MODES  STORAGECLASS  AGE
pvc-storage  Bound  pv-name 2Gi          RWO                                     5m
```

#### 5. Monte el volumen en un pod.

```
kubectl create -f pv-pod.yaml
```



Puede supervisar el progreso utilizando `kubectl get pod --watch`.

#### 6. Verifique que el volumen esté montado en `/my/mount/path`.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

#### 7. Ahora puede eliminar el Pod. La aplicación Pod ya no existirá, pero el volumen permanecerá.

```
kubectl delete pod pv-pod
```

Consulte el "[Objetos de Kubernetes y Trident](#)" para obtener más detalles sobre cómo interactúan las clases de almacenamiento con los `PersistentVolumeClaim` parámetros y para controlar la forma en que Trident aprovisiona los volúmenes.

## Expanda los volúmenes

Trident ofrece a los usuarios de Kubernetes la capacidad de expandir sus volúmenes una vez que se crean. Encuentre información sobre las configuraciones que se necesitan para ampliar los volúmenes iSCSI y NFS.

### Expanda un volumen iSCSI

Puede expandir un volumen persistente iSCSI (PV) mediante el aprovisionador CSI.



Los `solidfire-san` controladores , , `ontap-san-economy` admiten la expansión del volumen iSCSI `ontap-san` y requiere Kubernetes 1,16 y posterior.

## Paso 1: Configure el tipo de almacenamiento para que admita la ampliación de volumen

Edite la definición de StorageClass para definir `allowVolumeExpansion` el campo 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, edítelo para incluir el `allowVolumeExpansion` parámetro.

## Paso 2: Cree una RVP con el tipo de almacenamiento que ha creado

Edite la definición de PVC y actualice el `spec.resources.requests.storage` para reflejar el tamaño recién 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 (VP) y lo asocia con esta reclamación de volumen persistente (RVP).

```

kubect1 get pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound      pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi
RWO          ontap-san    8s

kubect1 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: Defina un pod que fije el PVC

Conecte el VP a un pod para que se cambie su tamaño. Existen dos situaciones a la hora de cambiar el tamaño de un VP iSCSI:

- Si el VP está conectado a un pod, Trident expande el volumen en el backend de almacenamiento, vuelve a escanear el dispositivo y cambia el tamaño del sistema de archivos.
- Cuando se intenta cambiar el tamaño de un VP no conectado, Trident expande el volumen en el back-end de almacenamiento. Una vez que la RVP está Unido a un pod, Trident vuelve a buscar el dispositivo y cambia el tamaño del sistema de archivos. Kubernetes, después, actualiza el tamaño de RVP después de completar correctamente la operación de ampliación.

En este ejemplo, se crea un pod que utiliza `san-pvc` el .

```
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: Expande el PV

Para cambiar el tamaño del VP que se ha creado de 1Gi a 2Gi, edite la definición de PVC y actualice 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: Validar la expansión

Puede validar correctamente la ampliación operativa comprobando el tamaño de la RVP, el VP y el volumen Trident:

```

kubect1 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
kubect1 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    |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

## Expanda un volumen NFS

Trident admite la expansión de volumen para VP NFS provisionados en `ontap-nas`, `ontap-nas-economy` `ontap-nas-flexgroup` `gcp-cvs` y `azure-netapp-files` back-ends.

### Paso 1: Configure el tipo de almacenamiento para que admita la ampliación de volumen

Para cambiar el tamaño de un PV de NFS, en primer lugar, el administrador debe configurar la clase de almacenamiento para permitir la expansión del volumen estableciendo `allowVolumeExpansion` el campo en `true`:

```

cat storageclass-ontapnas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true

```

Si ya creó un tipo de almacenamiento sin esta opción, puede editar el tipo de almacenamiento existente mediante el uso `kubect1 edit storageclass de` para permitir la expansión de volumen.

## Paso 2: Cree una RVP con el tipo de almacenamiento que ha creado

```
cat pvc-ontapnas.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Trident debe crear un PV de NFS de 20MiB TB para esta RVP:

```
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: Expande el PV

Para cambiar el tamaño del PV de 20MiB recién creado a 1GiB, edite la RVP y ajústelo `spec.resources.requests.storage` a 1GiB:

```
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: Validar la expansión**

Puede validar el tamaño correctamente trabajado comprobando el tamaño de la RVP, el VP y el volumen Trident:

```

kubect1 get pvc ontapnas20mb
NAME                STATUS      VOLUME
CAPACITY            ACCESS MODES  STORAGECLASS      AGE
ontapnas20mb       Bound       pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  1Gi
RWO                  ontapnas          4m44s

kubect1 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      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

## Importar volúmenes

Puede importar volúmenes de almacenamiento existentes como VP de Kubernetes mediante `tridentctl import`.

### Descripción general y consideraciones

Es posible importar un volumen en Trident para lo siguiente:

- Agrupe en contenedores una aplicación y vuelva a utilizar su conjunto de datos existente
- Utilice el clon de un conjunto de datos para una aplicación efímera
- Reconstruya un clúster de Kubernetes que haya fallado
- Migración de datos de aplicaciones durante la recuperación ante desastres

### Consideraciones

Antes de importar un volumen, revise las siguientes consideraciones.

- Trident solo puede importar volúmenes ONTAP de tipo RW (lectura y escritura). Los volúmenes del tipo DP (protección de datos) son volúmenes de destino de SnapMirror. Debe romper la relación de reflejo

antes de importar el volumen a Trident.

- Sugerimos importar volúmenes sin conexiones activas. Para importar un volumen que se usa activamente, clone el volumen y, a continuación, realice la importación.



Esto es especialmente importante en el caso de volúmenes de bloque, ya que Kubernetes no sabía que la conexión anterior y podría conectar fácilmente un volumen activo a un pod. Esto puede provocar daños en los datos.

- Aunque `StorageClass` debe especificarse en una RVP, Trident no utiliza este parámetro durante la importación. Durante la creación de volúmenes, se usan las clases de almacenamiento para seleccionar entre los pools disponibles según las características de almacenamiento. Como el volumen ya existe, no se requiere ninguna selección de pool durante la importación. Por lo tanto, la importación no fallará incluso si el volumen existe en un back-end o pool que no coincide con la clase de almacenamiento especificada en la RVP.
- El tamaño del volumen existente se determina y se establece en la RVP. Una vez que el controlador de almacenamiento importa el volumen, se crea el PV con un `ClaimRef` al PVC.
  - La política de reclamaciones se establece inicialmente en `retain` el VP. Una vez que Kubernetes enlaza correctamente la RVP y el VP, se actualiza la política de reclamaciones para que coincida con la política de reclamaciones de la clase de almacenamiento.
  - Si la política de reclamación de la clase de almacenamiento es `delete`, el volumen de almacenamiento se eliminará al eliminar el VP.
- De forma predeterminada, Trident administra la RVP y cambia el nombre de FlexVol y LUN en el back-end. Puede pasar `--no-manage` la marca para importar un volumen no gestionado. Si utiliza `--no-manage`, Trident no realiza ninguna operación adicional en la RVP o el VP durante el ciclo de vida de los objetos. El volumen de almacenamiento no se elimina cuando se elimina el VP, y también se ignoran otras operaciones como el clon de volumen y el cambio de tamaño de volumen.



Esta opción es útil si desea usar Kubernetes para cargas de trabajo en contenedores, pero de lo contrario desea gestionar el ciclo de vida del volumen de almacenamiento fuera de Kubernetes.

- Se agrega una anotación a la RVP y al VP que tiene el doble propósito de indicar que el volumen se importó y si se administran la PVC y la VP. Esta anotación no debe modificarse ni eliminarse.

## Importe un volumen

Puede usar `tridentctl import` para importar un volumen.

### Pasos

1. Cree el archivo de reclamación de volumen persistente (RVP) (por ejemplo, `pvc.yaml`) que se utilizará para crear la RVP. El archivo PVC debe incluir `name`, `namespace`, `accessModes` y `storageClassName`. Opcionalmente, puede especificar `unixPermissions` en la definición de RVP.

A continuación se muestra un ejemplo de una especificación mínima:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```



No incluya parámetros adicionales, como el nombre del VP o el tamaño del volumen. Esto puede provocar un error en el comando de importación.

2. Utilice `tridentctl import` el comando para especificar el nombre del back-end de Trident que contiene el volumen y el nombre que identifica de manera única el volumen en el almacenamiento (por ejemplo: ONTAP FlexVol, Element Volume, ruta Cloud Volumes Service). El `-f` argumento es necesario para especificar la ruta al archivo PVC.

```
tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-
file>
```

## Ejemplos

Revise los siguientes ejemplos de importación de volúmenes para los controladores compatibles.

### NAS de ONTAP y NAS FlexGroup de ONTAP

Trident admite la importación de volúmenes mediante `ontap-nas` los controladores y `ontap-nas-flexgroup`



- `ontap-nas-economy`El controlador no puede importar ni gestionar qtrees.`
- `ontap-nas`Los controladores y ontap-nas-flexgroup no permiten nombres de volúmenes duplicados.`

Cada volumen creado con el `ontap-nas` controlador es un FlexVol en el clúster de ONTAP. La importación de FlexVols con `ontap-nas` el controlador funciona igual. Una FlexVol que ya existe en un clúster de ONTAP puede importarse como `ontap-nas` una RVP. Del mismo modo, los volúmenes de FlexGroup se pueden importar como `ontap-nas-flexgroup` RVP.

### Ejemplos de NAS de ONTAP

A continuación, se muestra un ejemplo de un volumen gestionado y una importación de volumen no gestionada.

## Volumen gestionado

En el ejemplo siguiente se importa un volumen `managed_volume` llamado en un back-end llamado `ontap_nas`:

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
file	pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7	c5a6f6a4-b052-423b-80d4-8fb491a14a22	1.0 GiB	online	standard	true

## Volumen no gestionado

Cuando se utiliza `--no-manage` el 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
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
file	pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7	c5a6f6a4-b052-423b-80d4-8fb491a14a22	1.0 GiB	online	standard	false

## SAN de ONTAP

Trident admite la importación de volúmenes mediante `ontap-san` los controladores y `ontap-san-economy`

Trident puede importar volúmenes flexibles de SAN de ONTAP que contengan una única LUN. Esto es coherente con `ontap-san` el controlador, que crea una FlexVol para cada RVP y una LUN dentro de la FlexVol. Trident importa el FlexVol y lo asocia con la definición de PVC.

## Ejemplos de SAN de ONTAP

A continuación, se muestra un ejemplo de un volumen gestionado y una importación de volumen no gestionada.

### Volumen gestionado

Para los volúmenes gestionados, Trident cambia el nombre de FlexVol al `pvc-<uuid>` formato y a la LUN dentro de FlexVol a `lun0`.

El siguiente ejemplo importa el `ontap-san-managed` FlexVol que está presente en `ontap_san_default` el backend:

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
block	pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a	cd394786-ddd5-4470-adc3-10c5ce4ca757	20 MiB	online	basic	true

### Volumen no gestionado

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
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
block	pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228	e3275890-7d80-4af6-90cc-c7a0759f555a	1.0 GiB	online	san-blog	false

Si tiene LUN asignadas a iGroups que comparten un IQN con un IQN de nodo de Kubernetes, como se muestra en el ejemplo siguiente, recibirá el error: `LUN already mapped to initiator(s) in this group`. Deberá quitar el iniciador o desasignar la 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

### Elemento

Trident admite el software NetApp Element y la importación de volúmenes NetApp HCI mediante `solidfire-san` el controlador.



El controlador Element admite los nombres de volúmenes duplicados. Sin embargo, Trident devuelve un error si hay nombres de volúmenes duplicados. Como solución alternativa, clone el volumen, proporcione un nombre de volumen único e importe el volumen clonado.

### Ejemplo de elemento

En el siguiente ejemplo se importa un `element-managed` volumen en el back-end `element_default`.

```
tridentctl import volume element_default element-managed -f pvc-basic-import.yaml -n trident -d
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STORAGE CLASS	STATE	MANAGED
block	pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe	d3ba047a-ea0b-43f9-9c42-e38e58301c49	10 GiB	basic-element	online	true

### Google Cloud Platform

Trident admite la importación de volúmenes utilizando `gcp-cvs` el controlador.



Para importar un volumen respaldado por NetApp Cloud Volumes Service en Google Cloud Platform, identifique el volumen según la ruta de volumen. La ruta del volumen es la parte de la ruta de exportación del volumen después de `:/`. Por ejemplo, si la ruta de exportación es `10.0.0.1:/adroit-jolly-swift`, la ruta del volumen es `adroit-jolly-swift`.

### Ejemplo de Google Cloud Platform

En el ejemplo siguiente se importa `gcp-cvs` un volumen en el back-end `gcpcvs_YEppr` con la ruta de volumen de `adroit-jolly-swift`.

```
tridentctl import volume gcpcvs_YEppr adroit-jolly-swift -f <path-to-pvc-
file> -n trident
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
	pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55	e1a6e65b-299e-4568-ad05-4f0a105c888f	93 GiB	online	gcp-storage	true

## Azure NetApp Files

Trident admite la importación de volúmenes utilizando `azure-netapp-files` el controlador.



Para importar un volumen de Azure NetApp Files, identifique el volumen por su ruta de volumen. La ruta del 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:/importvoll1`, la ruta de volumen es `importvoll1`.

## Ejemplo de Azure NetApp Files

En el ejemplo siguiente se importa `azure-netapp-files` un volumen en el back-end `azurenappfiles_40517` con la ruta de volumen `importvoll1`.

```
tridentctl import volume azurenappfiles_40517 importvoll1 -f <path-to-
pvc-file> -n trident
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
file	pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab	1c01274f-d94b-44a3-98a3-04c953c9a51e	100 GiB	online	anf-storage	true

# Personalizar nombres y etiquetas de volúmenes

Con Trident puede asignar nombres y etiquetas significativos a los volúmenes que cree. Esto le ayuda a identificar y asignar fácilmente volúmenes a sus respectivos recursos de Kubernetes (RVP). También puede definir plantillas en el nivel de back-end para crear nombres de volúmenes y etiquetas personalizadas; los volúmenes que cree, importe o clone respetarán las plantillas.

## Antes de empezar

Nombres de volumen y etiquetas personalizables admiten:

1. Operaciones de creación, importación y clonado de volúmenes.
2. En el caso del controlador económico ontap-nas, solo el nombre del volumen Qtree debe cumplir con la plantilla de nombre.
3. En el caso del controlador ontap-san-economy, solo el nombre de la LUN cumple con la plantilla de nombre.

## Limitaciones

1. Los nombres de volumen personalizables solo son compatibles con los controladores locales de ONTAP.
2. Los nombres de volúmenes personalizables no se aplican a los volúmenes existentes.

## Comportamientos clave de los nombres de volúmenes personalizables

1. Si se produce un fallo debido a una sintaxis no válida en una plantilla de nombre, se produce un error en la creación del backend. Sin embargo, si la aplicación de plantilla falla, el volumen se nombrará de acuerdo con la convención de nomenclatura existente.
2. El prefijo de almacenamiento no es aplicable cuando se asigna el nombre de un volumen mediante una plantilla de nombres en la configuración back-end. Cualquier valor de prefijo deseado se puede agregar directamente a la plantilla.

## Ejemplos de configuración de backend con plantilla de nombre y etiquetas

Las plantillas de nombre personalizado se pueden definir en el nivel raíz y/o de grupo.

## 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.RequestName}}"
  },
  "labels": {"cluster": "ClusterA", "PVC":
    "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

## Ejemplo de 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 que considerar

1. En el caso de las importaciones de volúmenes, 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 gestionados, el nombre del volumen sigue a la plantilla de nombres definida en el nivel raíz en la definición de backend.
3. Trident no admite el uso de un operador de segmentos con el prefijo de almacenamiento.
4. Si las plantillas no dan como resultado nombres de volúmenes únicos, Trident añadirá algunos caracteres aleatorios para crear nombres de volúmenes únicos.
5. Si el nombre personalizado para un volumen económico NAS supera los 64 caracteres de longitud, Trident asignará un nombre a los volúmenes de acuerdo con la convención de nomenclatura existente. Para el resto de los controladores ONTAP, si el nombre del volumen supera el límite de nombre, se produce un error en el proceso de creación de volúmenes.

## Comparta un volumen NFS en espacios de nombres

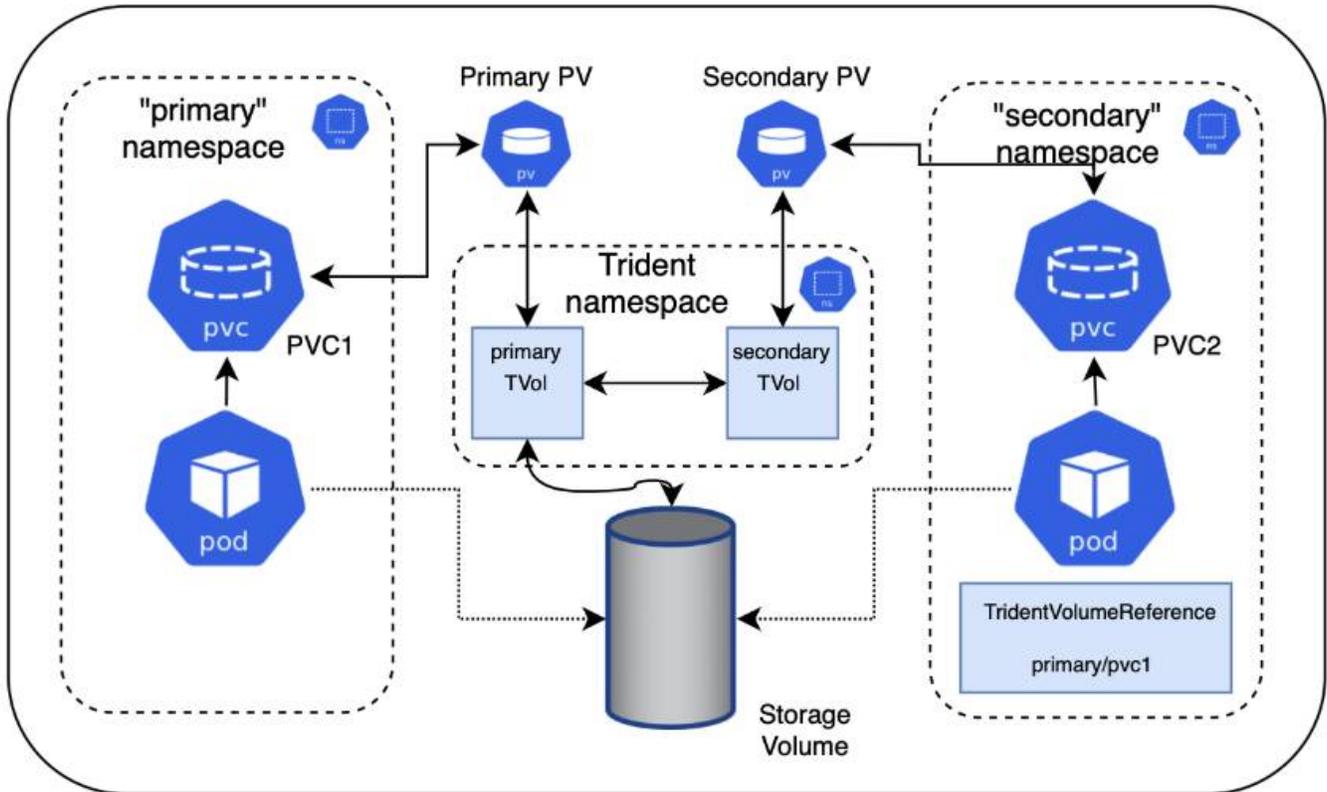
Con Trident, es posible crear un volumen en un espacio de nombres primario y compartirlo en uno o más espacios de nombres secundarios.

### Funciones

TridentVolumeReference CR permite compartir de forma segura los volúmenes NFS ReadWriteMany (RWX) en uno o varios espacios de nombres de Kubernetes. Esta solución nativa de Kubernetes tiene las siguientes ventajas:

- Varios niveles de control de acceso para garantizar la seguridad
- Funciona con todos los controladores de volúmenes NFS de Trident
- No depende de `tridentctl` ni de ninguna otra función de Kubernetes no nativa

Este diagrama ilustra el uso compartido de volúmenes de NFS en dos espacios de nombres de Kubernetes.



## Inicio rápido

Puede configurar el uso compartido del volumen NFS en unos pocos pasos.

1

### Configure la PVC de origen para compartir el volumen

El propietario del espacio de nombres de origen concede permiso para acceder a los datos de la RVP de origen.

2

### Otorgar permiso para crear una CR en el espacio de nombres de destino

El administrador del clúster concede permiso al propietario del espacio de nombres de destino para crear el sistema TridentVolumeReference CR.

3

### Cree TridentVolumeReference en el espacio de nombres de destino

El propietario del espacio de nombres de destino crea el TridentVolumeReference CR para hacer referencia al PVC de origen.

4

### Cree la RVP subordinada en el espacio de nombres de destino

El propietario del espacio de nombres de destino crea el PVC subordinado para utilizar el origen de datos desde el PVC de origen.

## Configurar los espacios de nombres de origen y destino

Para garantizar la seguridad, el uso compartido entre espacios de nombres requiere la colaboración y la acción del propietario del espacio de nombres de origen, el administrador de clúster y el propietario del espacio de nombres de destino. La función de usuario se designa en cada paso.

### Pasos

1. **Propietario del espacio de nombres de origen:** Crear el PVC (`pvc1`) en el espacio de nombres de origen que otorga permiso para compartir con el espacio de nombres de destino (`namespace2`) usando la `shareToNamespace` anotación.

```
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 VP y su volumen de almacenamiento NFS de back-end.



- Puede compartir el PVC en varios espacios de nombres utilizando una lista delimitada por comas. Por ejemplo, `trident.netapp.io/shareToNamespace: namespace2, namespace3, namespace4`.
- Puede compartir en todos los espacios de nombres utilizando `*`. Por ejemplo: `trident.netapp.io/shareToNamespace: *`
- Puede actualizar la RVP para incluir la `shareToNamespace` anotación en cualquier momento.

2. **Administrador de clúster:** cree la función personalizada y kubeconfig para conceder permiso al propietario del espacio de nombres de destino para crear el sistema `TridentVolumeReference` CR en el espacio de nombres de destino.
3. **Propietario del espacio de nombres de destino:** Crear un CR de `TridentVolumeReference` en el espacio de nombres de destino que se refiere 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:** Crear un PVC (namespace2)(pvc2 en el espacio de nombres de destino ) Utilizando la `shareFromPVC` anotación 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

```



El tamaño del PVC de destino debe ser menor o igual que el PVC de origen.

## Resultados

Trident lee la `shareFromPVC` anotación en la RVP de destino y crea el VP de destino como volumen subordinado sin ningún recurso de almacenamiento propio que apunte al VP de origen y comparta el recurso de almacenamiento VP de origen. La RVP y el VP de destino aparecen vinculados como normales.

## Elimine un volumen compartido

Es posible eliminar un volumen que se comparte en varios espacios de nombres. Trident eliminará el acceso al volumen en el espacio de nombres de origen y mantendrá el acceso a otros espacios de nombres que compartan el volumen. Cuando se eliminan todos los espacios de nombres que hacen referencia al volumen, Trident lo elimina.

## Se utiliza `tridentctl get` para consultar los volúmenes subordinados

Con `tridentctl` la utilidad, se puede ejecutar `get` el comando para obtener volúmenes subordinados. Para obtener más información, consulte [LINK:../Trident-reference/tridentctl.html](#) `tridentctl` commands and

options].

Usage:

```
tridentctl get [option]
```

Indicadores:

- `-h`, `--help`: Ayuda para volúmenes.
- `--parentOfSubordinate string`: Limite la consulta al volumen fuente subordinado.
- `--subordinateOf string`: Limite la consulta a los subordinados de volumen.

## Limitaciones

- Trident no puede evitar que los espacios de nombres de destino se escriban en el volumen compartido. Se debe usar el bloqueo de archivos u otros procesos para evitar la sobrescritura de datos de volúmenes compartidos.
- No puede revocar el acceso a la PVC de origen eliminando `shareToNamespace` las anotaciones o `shareFromNamespace` eliminando la `TridentVolumeReference` CR. Para revocar el acceso, debe eliminar el PVC subordinado.
- Las snapshots, los clones y el mirroring no son posibles en los volúmenes subordinados.

## Si quiere más información

Para obtener más información sobre el acceso de volúmenes entre espacios de nombres:

- Visite "[Uso compartido de volúmenes entre espacios de nombres: Dé la bienvenida al acceso al volumen entre espacios de nombres](#)".
- Vea la demostración en "[NetAppTV](#)".

## Replicar volúmenes mediante SnapMirror

Trident admite las relaciones de mirroring entre un volumen de origen en un clúster y el volumen de destino en el clúster con relación entre iguales para replicar datos para la recuperación ante desastres. Puede utilizar una definición de recursos personalizados (CRD) con nombre para realizar las siguientes operaciones:

- Crear relaciones de mirroring entre volúmenes (RVP)
- Elimine las relaciones de reflejo entre volúmenes
- Rompa las relaciones de reflejo
- Promocionar el volumen secundario durante condiciones de desastre (conmutaciones al respaldo).
- Realice una transición de las aplicaciones sin pérdidas de un clúster a otro (durante las migraciones y las conmutaciones al respaldo planificadas).

## Requisitos previos de replicación

Asegúrese de que se cumplen los siguientes requisitos previos antes de comenzar:

### Clústeres ONTAP

- **Trident:** La versión 22,10 o posterior de Trident debe existir tanto en los clústeres de Kubernetes de origen como de destino que utilizan ONTAP como backend.
- **Licencias:** Las licencias asíncronas de SnapMirror de ONTAP que utilizan el paquete de protección de datos deben estar habilitadas en los clústeres de ONTAP de origen y de destino. Consulte ["Información general sobre las licencias de SnapMirror en ONTAP"](#) si desea obtener más información.

### Interconexión

- **Cluster y SVM:** Los back-ends de almacenamiento ONTAP deben ser peered. Consulte ["Información general sobre relaciones entre iguales de clústeres y SVM"](#) si desea obtener más información.



Compruebe que los nombres de las SVM utilizados en la relación de replicación entre dos clústeres de ONTAP sean únicos.

- **Trident y SVM:** Las SVM remotas entre iguales deben estar disponibles para Trident en el clúster de destino.

### Controladores compatibles

- La replicación de volúmenes es compatible con los controladores ontap-nas y ontap-san.

## Cree una RVP reflejada

Siga estos pasos y utilice los ejemplos de CRD para crear una relación de reflejo entre los volúmenes primario y secundario.

### Pasos

1. Realice los siguientes pasos en el clúster de Kubernetes principal:
  - a. Cree un objeto StorageClass con el `trident.netapp.io/replication: true` parámetro.

### 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. Cree una RVP con el tipo de almacenamiento creado anteriormente.

### Ejemplo

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

- c. Cree 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 recupera la información interna del volumen y el estado actual de protección de datos (DP) del volumen y, a continuación, rellena el campo de estado del MirrorRelationship.

- d. Obtenga el TridentMirrorRelationship CR para obtener el nombre interno y SVM de la 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. Realice los siguientes pasos en el clúster de Kubernetes secundario:

- a. Cree una 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. Cree un CR de MirrorRelationship con información de destino y 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 de SnapMirror con el nombre de la política de relaciones configurada (o por defecto para ONTAP) e inicializará la misma.

- c. Crear una RVP con StorageClass creado anteriormente para que actúe como secundario (destino de SnapMirror).

### 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 el CRD de TridentMirrorRelationship y no podrá crear el volumen si la relación no existe. Si existe la relación, Trident se asegurará de que el nuevo FlexVol volume se coloque en una SVM relacionada con la SVM remota definida en MirrorRelationship.

## Estados de replicación de volúmenes

Una relación de mirroring de Trident (TMR) es un CRD que representa un extremo de una relación de replicación entre RVP. El TMR de destino tiene un estado que indica a Trident cuál es el estado deseado. El 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 anteriormente en esa relación de espejo.
  - El estado reestablecido se debe usar si el volumen de destino alguna vez mantuvo una relación con el volumen de origen debido a que sobrescribe el contenido del volumen de destino.
  - El estado reestablecido generará un error si el volumen no mantuvo una relación anteriormente con el origen.

## Promocione la RVP secundaria durante una conmutación al respaldo no planificada

Realice el siguiente paso en el clúster de Kubernetes secundario:

- Actualice el campo *spec.state* de TridentMirrorRelationship a *promoted*.

## Promocione la RVP secundaria durante una conmutación al respaldo planificada

Durante una conmutación al respaldo planificada (migración), realice los siguientes pasos para promocionar la RVP secundaria:

### Pasos

1. En el clúster de Kubernetes principal, cree una snapshot de la RVP y espere hasta que se cree la snapshot.
2. En el clúster de Kubernetes principal, cree SnapshotInfo CR para obtener información interna.

### 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, actualice el campo *spec.state* de *TridentMirrorRelationship* CR a *promoted* y *spec.promotedSnapshotHandle* para que sea *InternalName* de la snapshot.
4. En un clúster de Kubernetes secundario, confirme el estado (campo *status.state*) de *TridentMirrorRelationship* a *Promoted*.

## Restaurar una relación de mirroring después de una conmutación al nodo de respaldo

Antes de restaurar una relación de reflejo, elija el lado que desea realizar como el nuevo primario.

### Pasos

1. En el clúster de Kubernetes secundario, compruebe que se actualicen los valores del campo *spec.remoteVolumeHandle* del *TridentMirrorRelationship*.
2. En el clúster de Kubernetes secundario, actualice el campo *spec.mirror* de *TridentMirrorRelationship* a *reestablished*.

## Operaciones adicionales

Trident admite las siguientes operaciones en los volúmenes primarios y secundarios:

### Replica la PVC primaria a una nueva PVC secundaria

Asegúrese de que ya tiene un PVC primario y un PVC secundario.

### Pasos

1. Elimine los CRD de *PersistentVolumeClaim* y *TridentMirrorRelationship* del clúster secundario (destino) establecido.
2. Elimine el CRD de *TridentMirrorRelationship* del clúster primario (origen).
3. Cree un nuevo CRD de *TridentMirrorRelationship* en el clúster primario (de origen) para la nueva PVC secundaria (de destino) que desea establecer.

## Cambie el tamaño de una RVP reflejada, primaria o secundaria

El PVC se puede cambiar de tamaño como normal, ONTAP expandirá automáticamente cualquier flexvol de destino si la cantidad de datos excede el tamaño actual.

## Elimine la replicación de una RVP

Para eliminar la replicación, realice una de las siguientes operaciones en el volumen secundario actual:

- Elimine el `MirrorRelationship` en la RVP secundaria. Esto interrumpe la relación de replicación.
- O bien, actualice el campo `spec.state` a *Promoted*.

## Eliminar una RVP (que se había duplicado previamente)

Trident comprueba si hay PVR replicadas y libera la relación de replicación antes de intentar eliminar el volumen.

## Eliminar un TMR

La eliminación de un TMR en un lado de una relación reflejada hace que el TMR restante pase al estado *promocionado* antes de que Trident complete la eliminación. Si el TMR seleccionado para la eliminación ya se encuentra en el estado *promocionado*, no existe ninguna relación de reflejo y el TMR se eliminará y Trident promoverá la RVP local a *ReadWrite*. Esta eliminación libera los metadatos de `SnapMirror` del volumen local en ONTAP. Si este volumen se utiliza en una relación de reflejo en el futuro, debe utilizar un nuevo TMR con un estado de replicación de volumen *established* al crear la nueva relación de reflejo.

## Actualice las relaciones de reflejo cuando el ONTAP esté en línea

Las relaciones de reflejos se pueden actualizar en cualquier momento una vez establecidas. Puede utilizar los `state: promoted` campos o `state: reestablished` para actualizar las relaciones. Al promocionar un volumen de destino a un volumen de *ReadWrite* normal, se puede usar *promotedSnapshotHandle* para especificar una snapshot específica a la que restaurar el volumen actual.

## Actualice las relaciones de reflejo cuando la ONTAP esté sin conexión

Puede utilizar un CRD para realizar una actualización de `SnapMirror` sin que Trident tenga conectividad directa al clúster de ONTAP. Consulte 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 CRD `TridentActionMirrorUpdate`. Puede tomar un valor de *succeeded*, *in progress* o *failed*.

# Utilice Topología CSI

Trident puede crear y conectar volúmenes de forma selectiva a los nodos presentes en un clúster de Kubernetes utilizando el ["Función de topología CSI"](#).

## Descripción general

Con la función de topología CSI, el acceso a los volúmenes puede limitarse a un subconjunto de nodos, en función de regiones y zonas de disponibilidad. En la actualidad, los proveedores de cloud permiten a los administradores de Kubernetes generar nodos basados en zonas. Los nodos se pueden ubicar 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 la topología CSI.



Obtenga más información sobre la función Topología de CSI ["aquí"](#) .

Kubernetes ofrece dos modos de enlace de volúmenes únicos:

- Con `VolumeBindingMode` Establecer en `Immediate`, Trident crea el volumen sin reconocimiento de topología. La vinculación de volúmenes y el aprovisionamiento dinámico se manejan cuando se crea la RVP. Este es el valor por defecto `VolumeBindingMode` y es adecuado para clusters que no aplican restricciones de topología. Los volúmenes persistentes se crean sin depender de los requisitos de programación del pod solicitante.
- Con `VolumeBindingMode` establecido en `WaitForFirstConsumer`, la creación y vinculación de un volumen persistente para una RVP se retrasa hasta que se programe y cree un pod que utilice la RVP. De esta forma, se crean volúmenes con el fin de cumplir las restricciones de programación que se aplican en los requisitos de topología.



El modo de enlace `WaitForFirstConsumer` no requiere etiquetas de topología. Esto se puede utilizar independientemente de la característica de topología CSI.

## Lo que necesitará

Para utilizar la topología CSI, necesita lo siguiente:

- Un clúster de Kubernetes que ejecuta un ["Compatible con la versión de Kubernetes"](#)

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedea99df",
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:"1e11e4a2108024935ecfcb2912226cedea99df",
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 introduzcan el reconocimiento de topología

(`topology.kubernetes.io/region`y `topology.kubernetes.io/zone`). Estas etiquetas **deben estar presentes en los nodos del cluster** antes de instalar Trident para que Trident tenga en cuenta 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","kuber
netes.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","kuber
netes.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","kuber
netes.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: Cree un backend con detección de topología

Los back-ends de almacenamiento de Trident se pueden diseñar para aprovisionar volúmenes de forma selectiva según las zonas de disponibilidad. Cada backend puede llevar un bloque opcional `supportedTopologies` que representa una lista de zonas y regiones soportadas. En el caso de `StorageClasses` que utilizan dicho back-end, solo se creará un volumen si lo solicita una aplicación programada en una región/zona admitida.

A continuación se muestra un ejemplo de definición de backend:

## 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

```
{
  "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"}
  ]
}
```



`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.

También puede definir `supportedTopologies` por pool de almacenamiento. Consulte 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, `region` las etiquetas y `zone` representan la ubicación del pool de almacenamiento. `topology.kubernetes.io/region` `topology.kubernetes.io/zone` y dicte dónde se pueden consumir los pools de almacenamiento.

## Paso 2: Defina las clases de almacenamiento que tienen en cuenta la topología

En función de las etiquetas de topología que se proporcionan a los nodos del clúster, se puede definir `StorageClase` para que contenga información de topología. Esto determinará los pools de almacenamiento que sirven como candidatos para las solicitudes de RVP y el subconjunto de nodos que pueden usar los volúmenes aprovisionados mediante Trident.

Consulte 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 anteriormente, `volumeBindingMode` se establece en `WaitForFirstConsumer`. Las RVP solicitadas con este tipo de almacenamiento no se verán en cuestión hasta que se mencionan en un pod. Y, `allowedTopologies` proporciona las zonas y la región que se van a utilizar. `netapp-san-us-east1`StorageClass` crea RVP en el ``san-backend-us-east1` backend definido anteriormente.

### Paso 3: Cree y utilice un PVC

Con el clase de almacenamiento creado y asignado a un back-end, ahora puede crear RVP.

Vea el ejemplo `spec` a continuación:

```

---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san
spec:
accessModes:
- ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1

```

La creación de una RVP con este manifiesto daría como resultado lo siguiente:

```

kubect1 create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubect1 get pvc
NAME          STATUS      VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san      Pending
2s
kubect1 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
  ----      -
Normal    WaitForFirstConsumer  6s   persistentvolume-controller
waiting
for first consumer to be created before binding

```

Para que Trident cree un volumen y lo enlace a la RVP, use la RVP en un pod. Consulte 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: voll
      persistentVolumeClaim:
        claimName: pvc-san
  containers:
    - name: sec-ctx-demo
      image: busybox
      command: [ "sh", "-c", "sleep 1h" ]
      volumeMounts:
        - name: voll
          mountPath: /data/demo
      securityContext:
        allowPrivilegeEscalation: false

```

Este podSpec indica a Kubernetes que programe el pod en los nodos que están presentes en us-east1 la región y que elija entre cualquier nodo que esté presente en las us-east1-a zonas o. us-east1-b

Consulte la siguiente salida:

```
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
```

## Actualice los back-ends que se van a incluir `supportedTopologies`

Los back-ends preexistentes se pueden actualizar para incluir una lista de `supportedTopologies` uso `tridentctl backend update`. Esto no afectará a los volúmenes que ya se han aprovisionado, y sólo se utilizarán en las siguientes CVP.

## Obtenga más información

- ["Gestione recursos para contenedores"](#)
- ["Selector de nodos"](#)
- ["Afinidad y anti-afinidad"](#)
- ["Tolerancias y taints"](#)

## Trabajar con instantáneas

Las snapshots de volúmenes de Kubernetes de Persistent Volumes (VP) permiten copias puntuales de volúmenes. Es posible crear una copia Snapshot de un volumen creado con Trident, importar una copia de Snapshot creada fuera de Trident, crear un volumen nuevo a partir de una copia de Snapshot existente y recuperar datos de volumen de copias de Snapshot.

### Descripción general

La instantánea de volumen es compatible con `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, y `azure-netapp-files` los conductores.

#### Antes de empezar

Debe tener un controlador de instantánea externo y definiciones de recursos personalizados (CRD) para trabajar con instantáneas. Esta es la responsabilidad del orquestador de Kubernetes (por ejemplo: Kubeadm, GKE, OpenShift).

Si su distribución de Kubernetes no incluye el controlador de instantáneas y los CRD, consulte [Implemente una controladora Snapshot de volumen](#).



No cree una controladora Snapshot si crea instantáneas de volumen bajo demanda en un entorno de GKE. GKE utiliza un controlador de instantáneas oculto integrado.

## Cree una copia de Snapshot de volumen

### Pasos

1. Cree un `VolumeSnapshotClass`. Para obtener más información, consulte "[VolumeSnapshotClass](#)".
  - Los `driver` puntos al controlador CSI de Trident.
  - `deletionPolicy` puede ser `Delete` o `Retain`. Cuando se establece en `Retain`, la instantánea física subyacente del clúster de almacenamiento se conserva incluso si se elimina el `VolumeSnapshot` objeto.

### 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. Crear una instantánea de una RVP existente.

### Ejemplos

- En este ejemplo, se crea una copia Snapshot de una RVP existente.

```
cat snap.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

- En este ejemplo, se crea un objeto Snapshot de volumen para una RVP denominada `pvc1` y el nombre de la Snapshot se establece en `pvc1-snap`. Una Snapshot de volumen es similar a una RVP y se asocia con `VolumeSnapshotContent` un objeto que representa la snapshot real.

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                AGE
pvc1-snap           50s
```

- Es posible identificar `VolumeSnapshotContent` el objeto de `pvc1-snap` la Snapshot de volumen describiéndolo. El `Snapshot Content Name` identifica el objeto `VolumeSnapshotContent` que sirve para esta snapshot. `Ready To Use``El parámetro indica que la snapshot se puede usar para crear una nueva RVP.

```
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
.
.
```

## Cree una RVP a partir de una snapshot de volumen

Puede usar `dataSource` para crear una RVP con una `VolumeSnapshot` llamada `<pvc-name>` como origen de los datos. Una vez creada la RVP, se puede conectar a un pod y utilizarla como cualquier otro PVC.



La RVP se creará en el mismo back-end que el volumen de origen. Consulte "[KB: La creación de una RVP a partir de una snapshot de RVP de Trident no se puede crear en un back-end alternativo](#)".

En el siguiente ejemplo se crea la RVP utilizando `pvc1-snap` como origen 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: pvcl-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io

```

## Importe una copia de Snapshot de volumen

Trident admite "[Proceso de snapshot aprovisionado previamente de Kubernetes](#)" para permitir que el administrador de clúster cree `VolumeSnapshotContent` un objeto e importe copias de Snapshot creadas fuera de Trident.

### Antes de empezar

Trident debe haber creado o importado el volumen principal de la snapshot.

### Pasos

1. **Cluster admin:** Crear un `VolumeSnapshotContent` objeto que haga referencia a la instantánea backend. De esta forma, se inicia el flujo de trabajo Snapshot en Trident.
  - Especifique el nombre de la instantánea de backend en annotations como `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
  - Especifique `<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 `ListSnapshots` llamada.



`<volumeSnapshotContentName>` No siempre puede coincidir con el nombre de instantánea de backend debido a restricciones de nomenclatura de CR.

### Ejemplo

En el siguiente ejemplo se crea un `VolumeSnapshotContent` objeto que hace referencia a la instantánea 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:** Crear el `VolumeSnapshot` CR que hace referencia al `VolumeSnapshotContent` objeto. Esto solicita acceso para utilizar `VolumeSnapshot` en un espacio de nombres determinado.

### Ejemplo

En el siguiente ejemplo se crea una `VolumeSnapshot` CR denominada `import-snap` que hace referencia a la `VolumeSnapshotContent` `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 se requiere acción):** El Snapshotter externo reconoce el recién creado `VolumeSnapshotContent` y ejecuta la `ListSnapshots` llamada. Trident crea el `TridentSnapshot`.
  - El dispositivo de instantáneas externo establece el `VolumeSnapshotContent` en `readyToUse` y el `VolumeSnapshot` en `true`.
  - Trident devuelve `readyToUse=true`.
4. **Cualquier usuario:** Crear un `PersistentVolumeClaim` para hacer referencia al nuevo `VolumeSnapshot`, donde el `spec.dataSource` nombre (o `spec.dataSourceRef`) es el nombre `VolumeSnapshot`.

## Ejemplo

En el siguiente ejemplo se crea una RVP que hace referencia a la `VolumeSnapshot` llamada `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
```

## Recuperar datos de volumen mediante copias Snapshot

El directorio de snapshots está oculto de forma predeterminada para facilitar la máxima compatibilidad de los volúmenes aprovisionados mediante los `ontap-nas` controladores y `ontap-nas-economy`. Permita que `.snapshot` el directorio recupere datos de snapshots directamente.

Use la interfaz de línea de comandos de ONTAP para restaurar un volumen en un estado registrado en una snapshot anterior.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```



Cuando se restaura una copia Snapshot, se sobrescribe la configuración de volúmenes existente. Se pierden los cambios que se hagan en los datos del volumen después de crear la copia Snapshot.

## Restauración de volumen sin movimiento a partir de una copia de Snapshot

Trident permite restaurar volumen rápida y in situ a partir de una snapshot mediante `TridentActionSnapshotRestore` (TASR) CR. Esta CR funciona como una acción imprescindible de Kubernetes y no persiste una vez que finaliza la operación.

Trident soporta la restauración de instantáneas en `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup` `azure-netapp-files`, `gcp-cvs`, `google-cloud-netapp-volumes` y `solidfire-san` los conductores.

## Antes de empezar

Debe tener una snapshot de volumen disponible y la RVP vinculada.

- Compruebe que el estado de la RVP es de enlace.

```
kubectl get pvc
```

- Compruebe que la copia de Snapshot de volumen esté lista para utilizarse.

```
kubectl get vs
```

## Pasos

1. Cree el CR de TASR. En este ejemplo, se crea una CR para la RVP `pvc1` y una instantánea de volumen `pvc1-snapshot`.



El TASR CR debe estar en un espacio de nombres donde exista la PVC y 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
```

1. Aplique el CR para restaurar a partir de 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 a partir de la copia Snapshot. Es posible verificar el estado de restauración de la Snapshot.

```
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: pvcl
    volumeSnapshotName: pvcl-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```



- En la mayoría de los casos, Trident no vuelve a intentar automáticamente la operación en caso de fallo. Deberá realizar la operación de nuevo.
- Es posible que el administrador deba conceder permiso al usuario de Kubernetes sin acceso de administrador para crear una CR TASR en su espacio de nombres de la aplicación.

## Eliminar un VP con snapshots asociadas

Cuando se elimina un volumen persistente con instantáneas asociadas, el volumen Trident correspondiente se actualiza a un “estado de eliminación”. Quite las copias de Snapshot de volumen para eliminar el volumen de Trident.

## Implemente una controladora Snapshot de volumen

Si su distribución de Kubernetes no incluye el controlador de snapshots y los CRD, puede implementarlos de la siguiente manera.

### Pasos

1. Crear CRD de snapshot 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. Cree la controladora Snapshot.

```
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
```



Si es necesario, abra `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` y actualice `namespace` su espacio de nombres.

## Enlaces relacionados

- ["Copias de Snapshot de volumen"](#)
- ["VolumeSnapshotClass"](#)

## Información de copyright

Copyright © 2025 NetApp, Inc. Todos los derechos reservados. Imprimido en EE. UU. No se puede reproducir este documento protegido por copyright ni parte del mismo de ninguna forma ni por ningún medio (gráfico, electrónico o mecánico, incluidas fotocopias, grabaciones o almacenamiento en un sistema de recuperación electrónico) sin la autorización previa y por escrito del propietario del copyright.

El software derivado del material de NetApp con copyright está sujeto a la siguiente licencia y exención de responsabilidad:

ESTE SOFTWARE LO PROPORCIONA NETAPP «TAL CUAL» Y SIN NINGUNA GARANTÍA EXPRESA O IMPLÍCITA, INCLUYENDO, SIN LIMITAR, LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN CONCRETO, CUYA RESPONSABILIDAD QUEDA EXIMIDA POR EL PRESENTE DOCUMENTO. EN NINGÚN CASO NETAPP SERÁ RESPONSABLE DE NINGÚN DAÑO DIRECTO, INDIRECTO, ESPECIAL, EJEMPLAR O RESULTANTE (INCLUYENDO, ENTRE OTROS, LA OBTENCIÓN DE BIENES O SERVICIOS SUSTITUTIVOS, PÉRDIDA DE USO, DE DATOS O DE BENEFICIOS, O INTERRUPTIÓN DE LA ACTIVIDAD EMPRESARIAL) CUALQUIERA SEA EL MODO EN EL QUE SE PRODUJERON Y LA TEORÍA DE RESPONSABILIDAD QUE SE APLIQUE, YA SEA EN CONTRATO, RESPONSABILIDAD OBJETIVA O AGRAVIO (INCLUIDA LA NEGLIGENCIA U OTRO TIPO), QUE SURJAN DE ALGÚN MODO DEL USO DE ESTE SOFTWARE, INCLUSO SI HUBIEREN SIDO ADVERTIDOS DE LA POSIBILIDAD DE TALES DAÑOS.

NetApp se reserva el derecho de modificar cualquiera de los productos aquí descritos en cualquier momento y sin aviso previo. NetApp no asume ningún tipo de responsabilidad que surja del uso de los productos aquí descritos, excepto aquello expresamente acordado por escrito por parte de NetApp. El uso o adquisición de este producto no lleva implícita ninguna licencia con derechos de patente, de marcas comerciales o cualquier otro derecho de propiedad intelectual de NetApp.

Es posible que el producto que se describe en este manual esté protegido por una o más patentes de EE. UU., patentes extranjeras o solicitudes pendientes.

LEYENDA DE DERECHOS LIMITADOS: el uso, la copia o la divulgación por parte del gobierno están sujetos a las restricciones establecidas en el subpárrafo (b)(3) de los derechos de datos técnicos y productos no comerciales de DFARS 252.227-7013 (FEB de 2014) y FAR 52.227-19 (DIC de 2007).

Los datos aquí contenidos pertenecen a un producto comercial o servicio comercial (como se define en FAR 2.101) y son propiedad de NetApp, Inc. Todos los datos técnicos y el software informático de NetApp que se proporcionan en este Acuerdo tienen una naturaleza comercial y se han desarrollado exclusivamente con fondos privados. El Gobierno de EE. UU. tiene una licencia limitada, irrevocable, no exclusiva, no transferible, no sublicenciable y de alcance mundial para utilizar los Datos en relación con el contrato del Gobierno de los Estados Unidos bajo el cual se proporcionaron los Datos. Excepto que aquí se disponga lo contrario, los Datos no se pueden utilizar, desvelar, reproducir, modificar, interpretar o mostrar sin la previa aprobación por escrito de NetApp, Inc. Los derechos de licencia del Gobierno de los Estados Unidos de América y su Departamento de Defensa se limitan a los derechos identificados en la cláusula 252.227-7015(b) de la sección DFARS (FEB de 2014).

## Información de la marca comercial

NETAPP, el logotipo de NETAPP y las marcas que constan en <http://www.netapp.com/TM> son marcas comerciales de NetApp, Inc. El resto de nombres de empresa y de producto pueden ser marcas comerciales de sus respectivos propietarios.