



Aprovisiona y gestiona volúmenes

Trident

NetApp
October 29, 2026

Tabla de contenidos

Aprovisiona y gestiona volúmenes	1
Aprovisiona un volumen de Trident	1
Descripción general	1
Crea el PVC	1
Mover un volumen	3
Requisitos	4
Cómo funciona	4
Mover un volumen	4
Amplía los volúmenes de Trident	7
Expandir un volumen iSCSI	7
Expandir un volumen FC	11
Expandir un volumen NFS	14
Entiende los límites del subsistema RWX NVMe	17
Entiende el límite de 64 nodos	17
Entiende los modelos de subsistemas NVMe	17
Identifica los síntomas de error	18
Soluciona errores de límite de subsistema	18
Actualiza Trident para aplicar el modelo de super-subsystem	18
Escalabilidad del controlador	18
Conceptos clave y definiciones	19
Comportamiento predeterminado (modo serie)	19
Compatibilidad con la escalabilidad del controlador	19
Cómo funciona enableConcurrency	20
Antes de activar	20
Habilita la escalabilidad del controlador	20
Comportamiento de la concurrencia	22
Advertencias y limitaciones	23
Recomendaciones	23
Ampliación automática de volúmenes	23
Requisitos	24
Limitaciones	24
Provisiona volúmenes con la política de crecimiento automático	24
Crear una política de Autogrow	25
Crear una política de Autogrow	25
Política establece	26
Asocia una política con un StorageClass	26
Precedencia de políticas	27
Ejemplos de configuración	28
Gestiona las políticas de Autogrow	30
Ver políticas de Autogrow	30
Actualiza una política de Autogrow	31
Eliminar una política de Autogrow	31
Supervisa el uso de la política Autogrow	32

Protocolos compatibles	33
Backends compatibles	33
Limitaciones conocidas	33
Preguntas frecuentes	33
Importar volúmenes de Trident	39
Descripción general y consideraciones	39
Importar un volumen	40
Ejemplos	42
Ejemplos de ONTAP SAN-economy	47
Personalizar los nombres y las etiquetas de los volúmenes de Trident	51
Antes de empezar	51
Limitaciones	51
Comportamientos clave de los nombres de volúmenes personalizables	51
Ejemplos de configuración de backend con plantilla de nombre y etiquetas	51
Ejemplos de plantillas de nombres	53
Puntos a considerar	53
Compartir un volumen Trident NFS entre espacios de nombres	53
Funciones	53
Inicio rápido	54
Configura los espacios de nombres de origen y destino	55
Eliminar un volumen compartido	56
Usa <code>tridentctl get</code> para consultar volúmenes subordinados	56
Limitaciones	57
Para más información	57
Clonar volúmenes Trident entre espacios de nombres	57
Prerrequisitos	57
Inicio rápido	57
Configura los espacios de nombres de origen y destino	58
Limitaciones	59
Replica volúmenes usando SnapMirror	59
Requisitos previos de replicación	60
Crear un PVC reflejado	61
Estados de replicación de volúmenes	64
Promociona el PVC secundario durante una conmutación por error no planificada	64
Promociona el PVC secundario durante una conmutación por error planificada	64
Restaura una relación de réplica tras una conmutación por error	65
Operaciones adicionales	65
Actualizar las relaciones de réplica cuando ONTAP está en línea	66
Actualiza las relaciones de réplica cuando ONTAP está fuera de línea	66
Usa la topología CSI	66
Descripción general	66
Paso 1: crea un backend que tenga en cuenta la topología	68
Paso 2: Define StorageClasses que son conscientes de la topología	70
Paso 3: crea y utiliza un PVC	70
Actualizar los backends para incluir <code>supportedTopologies</code>	73

Encuentra más información	73
Trabajar con instantáneas de Trident.	73
Descripción general	73
Crear una instantánea de volumen	73
Crear una PVC a partir de una instantánea de volumen	75
Importar una instantánea de volumen	76
Recupera datos de volumen usando instantáneas	78
Restauración de volumen en el lugar desde una instantánea	78
Eliminar un PV con instantáneas asociadas	80
Implementa un controlador de instantáneas de volumen	80
Enlaces relacionados	81
Trabajar con instantáneas de grupos de volúmenes de Trident	81
Crear instantáneas de grupo de volúmenes	81
Recupera datos de volumen usando una instantánea de grupo	83
Restauración de volumen en el lugar desde una instantánea	83
Eliminar un PV con instantáneas de grupo asociadas.	83
Implementa un controlador de instantáneas de volumen	84
Enlaces relacionados	84

Aprovisiona y gestiona volúmenes

Aprovisiona un volumen de Trident

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 la 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
```



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.

Mover un volumen

A partir de Trident 26.06, un administrador de almacenamiento puede trasladar sin

interrupciones un volumen iSCSI a otro nodo ONTAP y a otro agregado dentro del mismo clúster ONTAP, mientras la carga de trabajo que lo utiliza sigue ejecutándose. Solicitas la migración de forma declarativa con un `TridentVolumeMove` recurso personalizado o el `tridentctl` equivalente, y Trident coordina ONTAP y el host para que la E/S nunca se interrumpa.

Requisitos

La función de desplazamiento de volúmenes se aplica solo al controlador `ontap-san` para volúmenes iSCSI. No se admiten los protocolos NVMe ni NAS.

Antes de mover un volumen, asegúrate de que se cumplan los siguientes requisitos:

- El backend usa el controlador `ontap-san` con la API de REST de ONTAP.
- El clúster ONTAP tiene al menos dos nodos.
- Dispones de uno de los siguientes conjuntos de credenciales:
 - ONTAP 9.19.1 o posterior si usas la función personalizada dedicada de ONTAP.
 - Credenciales de Cluster-admin en versiones anteriores de ONTAP.

Cómo funciona

Trident traslada el volumen (LUN) entre dos ubicaciones dentro del mismo clúster de ONTAP. Antes de que se trasladen los datos, Trident añade nuevas rutas iSCSI al destino junto a las rutas existentes, de modo que el dispositivo multivía del host nunca pierda el acceso. Una vez que ONTAP ha completado el traslado, Trident elimina las rutas de origen obsoletas.

Trident gestiona el traslado a través de una secuencia ordenada de estados que se pueden consultar en el recurso `TridentVolumeMove`, de modo que puedas seguir el progreso del traslado.

Mover un volumen

Identifica el volumen y las ubicaciones

1. Busca el nombre del PV que quieres mover. El nombre del PV es también el nombre del recurso de movimiento.
2. Desde el clúster ONTAP, anota el nodo y el agregado actuales (de origen) de ese volumen y elige el nodo y el agregado de destino.

Empieza la mudanza

Utiliza cualquiera de los dos métodos. Ambos crean el mismo `TridentVolumeMove` recurso, cuyo nombre es el del PV.

Usando `tridentctl`

```
tridentctl create volume-move \  
  --volume      pvc-8a814d62-bd58-4253-b0d1-82f2885db671 \  
  --source-node ontap-node-01 \  
  --target-node  ontap-node-02
```

```
--source-pool aggr1_node01 \  
--target-node ontap-node-02 \  
--target-pool aggr1_node02 \  
-n trident
```

tvm es un alias de volume-move. Agrega `--delete-after-success 10m` para eliminar el recurso automáticamente después de que el traslado se complete con éxito.

Uso de un recurso TridentVolumeMove

```
apiVersion: trident.netapp.io/v1  
kind: TridentVolumeMove  
metadata:  
  name: pvc-8a814d62-bd58-4253-b0d1-82f2885db671 # the PV to move  
  namespace: trident  
spec:  
  sourceNode: ontap-node-01  
  sourcePool: aggr1_node01  
  targetNode: ontap-node-02  
  targetPool: aggr1_node02  
  # deleteAfterSuccess: 10m # optional
```

Aplica el recurso:

```
kubectl apply -f tridentvolumemove.yaml
```

Campos de especificaciones

sourceNode / targetNode

Nombres de los nodos ONTAP (no UUIDs).

sourcePool / targetPool

Nombres de agregados de ONTAP.

deleteAfterSuccess

Duración opcional, como 30s o 10m. Establece el valor en 0 para eliminar el recurso inmediatamente, u omítelo para conservar el recurso.

Supervisa la mudanza

Sigue el progreso usando el `tvm` nombre abreviado o `tridentvolumemove`:

```
kubectl get tridentvolumemove -n trident
```

Para ver el estado actual y el progreso por nodo, describe el recurso:

```
kubectl describe tridentvolumemove <pv-name> -n trident
```

El volumen puede tener conexiones simultáneas durante el estado `Moving`, porque ONTAP puede tardar un tiempo en completar el traslado de un volumen de gran tamaño. Para garantizar un traslado sin interrupciones, Trident coordina entre el controlador Trident y el nodo Trident. Cada estado realiza los siguientes pasos en orden, terminando en `Succeeded` o `Failed`.

Estado	Qué está pasando
Pending	Trident acepta y valida la solicitud de traslado y recopila la información necesaria antes de que se inicien los cambios en el almacenamiento.
ControllerStaging	Trident prepara la ubicación de destino. Asigna las interfaces de almacenamiento del nodo de destino al LUN y actualiza Kubernetes para que un futuro reinicio del nodo se vuelva a conectar a la ubicación correcta.
NodeStaging	En cada host que tenga el volumen montado, Trident añade las nuevas rutas de almacenamiento del nodo de destino junto a las rutas existentes, de modo que las operaciones de E/S del pod continúen a través de ambos conjuntos de rutas.
Moving	ONTAP copia y transfiere los datos del volumen al agregado de destino. Esta es la fase más larga, y ONTAP se encarga de todo mientras Trident realiza sondeos hasta que termina. Durante este estado se permiten nuevas publicaciones.
ControllerUnstaging	La transferencia de datos ha finalizado. Trident actualiza Kubernetes para eliminar las rutas de origen del estado deseado y luego desvincula las interfaces de almacenamiento del nodo de origen del LUN.
NodeUnstaging	En cada host, Trident elimina las rutas de origen obsoletas. El host solo conserva las rutas de destino y el dispositivo multivía sigue gestionando las operaciones de E/S sin interrupción.
Succeeded	El volumen se ha trasladado por completo y ya está disponible en línea en el destino. Trident conserva el <code>TridentVolumeMove</code> recurso, o lo elimina automáticamente si configuras <code>deleteAfterSuccess</code> .
Failed	Un error irreparable detuvo la operación de movimiento o impidió que comenzara. Trident restablece el estado del volumen para que las operaciones normales puedan reanudarse y mantiene el recurso con un mensaje de error. Revisa el mensaje, soluciona la causa raíz, elimina el recurso y vuelve a intentarlo.

La `.status.attachments` lista muestra el progreso de cada nodo adjunto en su campo `state`:

Estado de adjunto	Significado
Pending	Nodo identificado como conectado; aún no se ha iniciado el trabajo.
Bridged	Se han añadido nuevas rutas de destino junto a las rutas existentes.
Migrated	El tráfico ahora se sirve por las nuevas rutas.
Cleaned	Se eliminaron las rutas de origen obsoletas del nodo.

Estado de adjunto	Significado
Detached	El nodo se desvinculó a mitad del movimiento y se excluyó de cualquier consideración posterior mientras Trident evalúa si la vinculación está lista para pasar al siguiente paso.
Failed	Error de terminal en este nodo. Consulta el mensaje.

Limpia el recurso de traslado

Si configuras `deleteAfterSuccess`, Trident elimina el recurso automáticamente. De lo contrario, puedes eliminarlo manualmente. Antes de solicitar otro traslado en el mismo volumen, asegúrate de que no exista ningún recurso `TridentVolumeMove` para ese volumen en estado `Succeeded` o `Failed`.



Trident nunca elimina automáticamente las operaciones fallidas. Trident conserva un `Failed` recurso como registro de auditoría. Una vez que hayas solucionado la causa raíz, elimina el recurso manualmente antes de volver a intentarlo.

Amplía los volúmenes de Trident

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.



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 |
+-----+-----+-----+-----+
|          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.



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, editálo 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
```

```
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
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 NVMe RWX no puede montarse 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-subsystema 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 mejora la escalabilidad del controlador gracias a una mayor concurrencia entre múltiples controladores de almacenamiento. Al habilitar la escalabilidad del controlador, el controlador de Trident procesa las operaciones de almacenamiento en paralelo en lugar de hacerlo en serie, lo que aumenta el rendimiento en entornos de Kubernetes con numerosas operaciones simultáneas.

Antes de realizar la implementación, comprueba qué controladores de Trident admiten la escalabilidad del controlador en su versión de disponibilidad general y cuáles están disponibles como vista previa técnica en Trident 26.06. Esto te ayuda a tomar decisiones de implementación fundamentadas y a gestionar el riesgo. La escalabilidad del controlador está desactivada de forma predeterminada.

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 tras un único bloqueo. Estas operaciones incluyen la creación, eliminación y cambio de tamaño de volúmenes; la creación y eliminación de instantáneas; la publicación y retirada de volúmenes; y la gestión del backend.

Al habilitar la escalabilidad del controlador, las operaciones en diferentes volúmenes y backends se ejecutan de forma simultánea. Esto aumenta el rendimiento y reduce el tiempo de ejecución de principio a fin en entornos con un elevado número de operaciones simultáneas de PersistentVolumeClaim y VolumeSnapshot.

Comportamiento predeterminado (modo serie)

De forma predeterminada, el controlador Trident procesa las operaciones una por una. Cada solicitud de creación, eliminación, cambio de tamaño o instantánea se completa antes de que Trident inicie la siguiente. El modo serie es el valor predeterminado compatible con todas las instalaciones y no requiere configuración alguna.

El modo serie es suficiente para la mayoría de las cargas de trabajo. Activa la escalabilidad del controlador solo cuando el volumen de operaciones genere un retraso en el procesamiento en serie.

Compatibilidad con la escalabilidad del controlador

Trident admite la escalabilidad de los controladores en distintos niveles de madurez para diferentes controladores de almacenamiento.

Disponibilidad general

Los siguientes drivers soportan la escalabilidad del controlador en disponibilidad general en Trident 26.06:

- `ontap-san`
- `ontap-nas`
- `ontap-nas-economy`
- `ontap-san-economy`
- `google-cloud-netapp-volumes`
- `azure-netapp-files`
- `solidfire-san`



Los controladores `google-cloud-netapp-volumes` y `google-cloud-netapp-volumes-san` son diferentes. Trident solo es compatible con `google-cloud-netapp-volumes`. No utilices `google-cloud-netapp-volumes-san` en las configuraciones del backend ni en los ejemplos.

Avance técnico

El siguiente controlador admite la escalabilidad del controlador como función de vista previa técnica en Trident 26.06:

- `asa-r2` (SAN y NVMe)

Este controlador presenta las siguientes limitaciones:

- La concurrencia de controladores está disponible únicamente para evaluación y pruebas.
- El comportamiento puede variar de una versión a otra.
- NetApp no recomienda su uso en entornos de producción.

Cómo funciona `enableConcurrency`

Cuando configuras `enableConcurrency` en `true`, Trident aplica el procesamiento simultáneo en todos los backends que gestiona el controlador. La configuración se aplica a todos los backends a la vez. No puedes habilitarlo para backends individuales ni para drivers individuales.

Cada backend configurado debe utilizar un controlador que figure en la lista de disponibilidad general o de vista previa técnica. Si algún backend utiliza un controlador no compatible, Trident no se inicia y Trident no añade un backend que utilice un controlador no compatible.

Para restablecer el arranque, elimina o reconfigura cualquier backend que utilice un controlador no compatible o desactiva la escalabilidad del controlador.

Antes de activar



Antes de habilitar la escalabilidad del controlador, comprueba que todos los backends configurados utilicen un controlador que figure en la lista de disponibilidad general o de vista previa técnica. Si algún backend utiliza un controlador no compatible, Trident no se inicia después de habilitar la función.

Utiliza la siguiente tabla para decidir si deseas habilitar la escalabilidad del controlador.

Si tu entorno	Entonces
Utiliza únicamente controladores compatibles y presenta un retraso en las operaciones del controlador	Habilita la escalabilidad del controlador.
Utiliza cualquier controlador no compatible	Mantén el modo serie predeterminado. No actives la escalabilidad del controlador.
Gestiona un volumen de operaciones bajo sin acumular retrasos	Mantén el modo serie predeterminado.

Habilita la escalabilidad del controlador

La opción de configuración `enableConcurrency` controla la escalabilidad del controlador. Debes habilitar esta opción de forma explícita durante la instalación de Trident o cuando actualices una implementación existente.

Despliegue del operador Trident

Para habilitar la escalabilidad del controlador con el operador Trident, configura `enableConcurrency` en `true` en el recurso personalizado (CR) `TridentOrchestrator`.

Nueva instalación

Crea o edita el TridentOrchestrator CR y establece enableConcurrency 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}}'
```

Comprueba que Trident haya aplicado la configuración:

```
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 con 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 existente `tridentctl`, desinstala Trident y vuelve a instalarlo con el parámetro:

```
tridentctl uninstall -n trident
tridentctl install -n trident --enable-concurrency
```

Comprueba que la escalabilidad del controlador esté activada

Una vez que habilites la escalabilidad del controlador, verifica que el controlador Trident se esté ejecutando con la concurrencia activada. Revisa los registros del pod del controlador:

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

La salida incluye una entrada en el registro que confirma que la concurrencia está habilitada.

Comportamiento de la concurrencia

Cuando se habilita la escalabilidad del controlador, el controlador Trident aplica el siguiente comportamiento:

- Trident sustituye el bloqueo global único por un sistema de bloqueo más detallado, específico para cada recurso.
- Trident serializa las operaciones que modifican el mismo recurso para mantener la coherencia de los datos.

- Las operaciones que solo leen de un recurso se ejecutan de forma concurrente con otras operaciones de lectura en ese mismo recurso.
- Trident limita a 20 el número de solicitudes simultáneas de la API de ONTAP por cada LIF de gestión para evitar la sobrecarga de los sistemas de almacenamiento backend.
- Si varios backends comparten el mismo LIF de gestión, comparten este límite de 20 solicitudes.

Advertencias y limitaciones

Las siguientes consideraciones se aplican a la escalabilidad de los controladores en Trident 26.06:

- La escalabilidad del controlador solo es compatible con los controladores que figuran en las listas de disponibilidad general y de vista previa técnica. Para obtener más información, consulta [Antes de activar](#).
- El controlador Trident gestiona la concurrencia de forma interna. Esta versión no ofrece límites de concurrencia configurables por el usuario.
- El rendimiento global depende del controlador de almacenamiento en uso, la capacidad de respuesta del backend y el rendimiento del servidor de la API de Kubernetes.
- Una elevada concurrencia puede aumentar la carga en los sistemas de almacenamiento del backend.
- El comportamiento de la escalabilidad del controlador no es idéntico en todos los drivers.
- El controlador de la versión preliminar técnica puede presentar un rendimiento irregular bajo cargas elevadas y su comportamiento puede variar entre versiones.
- La depuración de operaciones concurrentes puede resultar más compleja debido a la ejecución en paralelo. Las métricas y los registros pueden mostrar resultados de operaciones entremezclados.

Recomendaciones

Antes de habilitar la escalabilidad del controlador, completa los siguientes pasos:

1. Comprueba que todos los backends configurados utilicen un controlador que esté en la lista de disponibilidad general o de vista previa técnica.
2. Prueba el cambio en un clúster que no sea de producción antes de aplicarlo en el entorno de producción.
3. Comprueba que la escalabilidad del controlador esté activada después de aplicar el cambio.

Aplica las siguientes recomendaciones generales cuando trabajes con la escalabilidad del controlador:

- Utiliza controladores de disponibilidad general para entornos de producción que requieran alta escalabilidad.
- Prueba el controlador de la versión preliminar técnica únicamente en entornos que no sean de producción.
- Supervisa el rendimiento del backend y de los controladores cuando operas a gran escala.
- No des por sentado 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



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
usedThreshhold	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.



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.
trident.netapp.io/autogrowPolicy: standard-agp	trident.netapp.io/autogrowPolicy: logs-policy	Utiliza logs-policy (PVC anula StorageClass).

Anotación de StorageClass	Anotación de PVC	Comportamiento eficaz
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 omite `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"
```



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. También puedes supervisar qué volúmenes usan 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.



Los cambios afectan a todos los volúmenes que utilizan la política. Cuando sea posible, prueba primero los cambios en un entorno que no sea de producción.

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 los nuevos parámetros en la próxima 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 del finalizador para evitar el borrado accidental mientras los volúmenes las usan activamente.

Pasos

1. Elimina la política:

```
kubectl delete tridentautogrowpolicy production-db-policy
```

2. Si los volúmenes siguen utilizando la política, la eliminación pasa a un estado `Deleting`. Comprueba 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 utilizan la política	La política entra <code>Deleting</code> 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



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`.

Backends compatibles

Autogrow es compatible con los backends de ONTAP. Autogrow también es compatible con Google Cloud NetApp Volumes (GCNV) para volúmenes NAS y SAN.

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 (implementaciones en entornos ya existentes):** Es posible que la función de autogrow no funcione con los volúmenes existentes, incluso si se aplica una Autogrow Policy válida. Esta limitación se debe a una migración en curso de publicaciones de volúmenes. Para confirmar que la migración se ha completado, revisa los registros del controlador Trident en busca de 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 obtener 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?

Configura la anotación de PVC en "none". Esta configuración anula cualquier política a nivel de 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 ampliar el volumen. No genera más solicitudes de redimensionamiento para ese volumen, aunque el uso siga aumentando más allá de usedThreshold.

Por ejemplo, con esta política, Trident deja de ampliar el volumen una vez que alcanza 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 es necesario reiniciar ningún volumen.

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 de finalizadores. Si los volúmenes siguen utilizando la política, la eliminación pasa a un estado `Deleting` y queda en espera hasta que elimines todos los volúmenes 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 elimines todos los volúmenes, Trident libera el finalizador y elimina la política.

¿Qué backends admite la ampliación automática?

Autogrow es compatible con los backends de ONTAP. También es compatible con Google Cloud NetApp Volumes (GCNV) tanto para volúmenes NAS como SAN.

Autogrow es compatible con los protocolos NFS, iSCSI, FCP y NVMe. Los volúmenes de bloques sin procesar NVMe requieren ONTAP 9.16.1 o una versión posterior.

Es posible que los volúmenes existentes en implementaciones en entornos brownfield requieran que se complete la migración de la publicación de volúmenes antes de que el autogrow surta efecto. Para comprobar

el estado de la migración, consulta 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 de Trident

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.



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`, Trident elimina el volumen de almacenamiento cuando se elimina el PV.
- De forma predeterminada, Trident gestiona el PVC y cambia el nombre del volumen `FlexVol` y del LUN o `qtree` en el backend. Puedes pasar la `--no-manage` flag para importar un volumen no gestionado y la `--no-rename` flag 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 gestiona el ciclo de vida de los volúmenes. Esta opción solo es compatible con los controladores `ontap-nas`, `ontap-san` (incluidos los sistemas ASA r2), `ontap-san-economy` y `ontap-nas-economy` (importación de `qtree`).



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.



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
```



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 volume` 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.



- El `ontap-nas-economy` driver gestiona el almacenamiento a nivel de `qtree`, así que no admite importar un volumen `FlexVol` por sí solo. Para importar `qtrees` con el `ontap-nas-economy` driver, consulta [Importar qtrees](#) (Tech Preview). Cuando Trident importa un `qtree`, también importa el `FlexVol` principal como un bucket que contiene el `qtree`.
- 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 |
+-----+-----+-----+-----+
| 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 |
+-----+-----+-----+-----+
| 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>

```

Importar qtrees



La importación de qtree es una función en fase de Tech Preview de Trident 26.06. Las funciones en fase de Tech Preview no son compatibles para uso en producción.

A partir de Trident 26.06, es posible importar qtrees de ONTAP ya existentes a Trident usando el controlador ontap-nas-economy. Identifica el qtree por su volumen FlexVol y los nombres del qtree en formato flexvol/qtree, junto con el UUID del backend que lo contiene. Las consideraciones generales sobre la

importación en [Descripción general y consideraciones](#) también se aplican a la importación de qtrees.

Importar un qtree usando tridentctl

Especifica el qtree en formato flexvol/qtree como nombre del volumen. El siguiente ejemplo importa el qtree qtree1 desde el volumen FlexVol flexvol1 en el backend ontap_nas_economy:

```
tridentctl import volume ontap_nas_economy flexvol1/qtree1 -f <path-to-pvc-file>
```

Para importar el qtree sin cambiarle el nombre, añade el `--no-rename` flag. Para importar el qtree sin que Trident gestione su ciclo de vida, añade el `--no-manage` flag.

Importar un qtree usando anotaciones PVC

Crea un PVC con las siguientes anotaciones:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "flexvol1/qtree1"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: <size>
    storageClassName: <storage-class-name>
```

Para conservar los nombres existentes del volumen FlexVol y del qtree, añade la anotación `trident.netapp.io/importNoRename: "true"`. Para importar el qtree como no gestionado, añade la anotación `trident.netapp.io/notManaged: "true"`.

Nomenclatura y comportamiento de gestión

Importación gestionada (predeterminada)

Trident cambia el nombre del qtree para que coincida con el nombre del PVC y renombra el volumen FlexVol según la `ontap-nas-economy` convención de nomenclatura. La ruta de unión no cambia.

`--no-rename`

Trident conserva los nombres actuales de los volúmenes FlexVol y de las qtree. Los nombres ya deben seguir las convenciones de nomenclatura de Trident, o Trident no podrá gestionar las operaciones de eliminación o montaje de la qtree.

--no-manage

Trident hace referencia al qtree, pero no gestiona su ciclo de vida. Los nombres deben seguir las convenciones de nomenclatura de Trident, y Trident no utiliza ni gestiona políticas de exportación automática para el qtree.

Comportamiento de la política de exportación

Si el backend utiliza políticas de exportación automática, el qtree importado utiliza el modelo de políticas de exportación por qtree, en el que cada qtree tiene su propia política de exportación. Trident asigna al qtree una política de exportación vacía en el momento de la importación. En el momento de la publicación, Trident actualiza el volumen FlexVol con la política de exportación específica de cada backend si la exportación automática está habilitada y la política de exportación existente FlexVol es `default` o está vacía.



Actualizar la política de exportación del volumen FlexVol puede interrumpir otros qtrees en el mismo volumen FlexVol que estén montados actualmente. Importa los qtrees solo cuando los qtrees en el volumen FlexVol estén desmontados.



No apuntes dos clústeres de Kubernetes al mismo SVM activo, y no importes un qtree que otro clúster siga gestionando. Si ambos clústeres gestionan el mismo volumen, las operaciones conflictivas pueden dañar o destruir los datos.

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



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 por defecto 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`.



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.



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 azurenetaappfiles_40517 con la ruta de volumen importvol1.

```
tridentctl import volume azurenetaappfiles_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
|
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

Personalizar los nombres y las etiquetas de los volúmenes de Trident

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.RequestName}}"
},
"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.

Compartir un volumen Trident NFS entre espacios de nombres

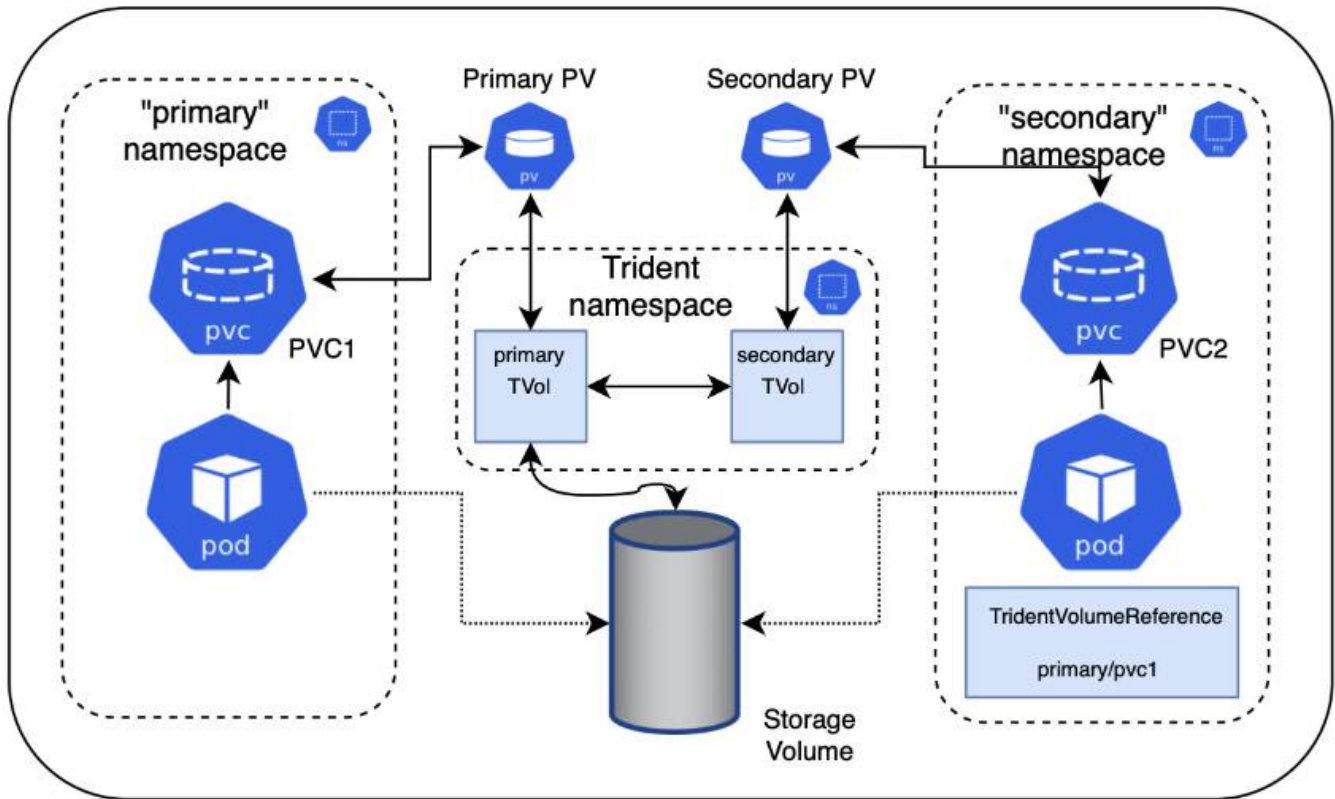
Con Trident, puedes crear un volumen en un espacio de nombres primario y compartirlo en uno o más espacios de nombres secundarios.

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.



- 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
```



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](https://docs.netapp.com/es/ontap/9.10/trident-reference/trident-cl.html#tridentctl-comandos-y-opciones) `[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 Trident entre 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.



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.



- 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 un CR de «`TridentVolumeReference`» en el espacio de nombres de destino que haga referencia al PVC 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 provisionados 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 solo archivo que habilita varias funciones. Consulta ["Licencias incluidas con ONTAP One"](#) para obtener más información.



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.



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)



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.

```
kubect1 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
  replicationPolicy: MirrorAllSnapshots
  replicationSchedule: 5min
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
      "datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

El TMR spec cuenta con cuatro campos de nivel superior: `state`, que establece el estado de relación deseado (consulta [Estados de replicación de volúmenes](#)); `volumeMappings`, que asocia los PVC locales con sus identificadores de volumen remotos; y los campos opcionales `replicationPolicy` y `replicationSchedule`, que se describen a continuación.

Los campos `replicationPolicy` y `replicationSchedule` son opcionales. Establece `replicationPolicy` con el nombre de una política asíncrona de SnapMirror en ONTAP y `replicationSchedule` con el nombre de una programación de ONTAP. Si omites cualquiera de los campos, Trident usa la política de replicación o la programación correspondientes configuradas en el backend. Establece estos campos cuando creas la relación; no puedes cambiar la política ni la programación después de que se establezca la relación. Para usar una configuración diferente, elimina y vuelve a crear el TMR.

Trident crea e inicializa una relación SnapMirror utilizando la política y la programación de replicación configuradas.

1. 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 principal de Kubernetes, crea el `TridentTridentSnapshotInfoCR` para obtener información interna.

Ejemplo

```
kind: TridentSnapshotInfo
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  snapshotName csi-nas-snapshot
```

- a. En el clúster secundario de Kubernetes, actualiza el campo `spec.state` del CR `TridentMirrorRelationship` a `promoted` y `spec.volumeMappings[].promotedSnapshotHandle` para que sea el `internalName` de la instantánea.

3. 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 secundario de Kubernetes, actualiza el campo `spec.state` 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.



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.



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"
[nodel1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"nodel1","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","kubernetes.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","kubernetes.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-b"}]
```

```
kubernetes.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

```
---
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.

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
  ----      -
Normal    WaitForFirstConsumer  6s    persistentvolume-controller
waiting
for first consumer to be created before binding
Message
```

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: 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 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

```

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"](#)

Trabajar con instantáneas de Trident

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](#).



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 pvcl-snap
Name:          pvcl-snap
Namespace:     default
...
Spec:
  Snapshot Class Name:  pvcl-snap
  Snapshot Content Name: snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:      PersistentVolumeClaim
    Name:      pvcl
  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.



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 `pvcl-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`.



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
```



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`.



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: ""
```



- 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.yaml
1
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
```



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"](#)

Trabajar con instantáneas de grupos de volúmenes de Trident

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.



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`



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](#).



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

```



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 obtener 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_volumegroupsnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcontents.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
```



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"](#)

Información de copyright

Copyright © 2026 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 INTERRUPCIÓN 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.