Provisión y gestión de volúmenes

Trident

NetApp
January 15, 2026

Tabla de contenidos

Provisión y gestión de volúmenes . . . 1
- Provisión de un volumen . . . 1
  - Descripción general . . . 1
  - Crea el PVC . . . 1
- Ampliar volúmenes . . . 4
  - Expandir un volumen iSCSI . . . 4
  - Ampliar un volumen FC . . . 8
  - Expandir un volumen NFS . . . 12
- volúmenes de importación . . . 15
  - Descripción general y consideraciones . . . 15
  - Importar un volumen . . . 16
  - Ejemplos . . . 17
- Personaliza los nombres y etiquetas de los volúmenes. . . . 23
  - Antes de empezar . . . 23
  - Limitaciones . . . 23
  - Comportamientos clave de los nombres de volumen personalizables . . . 23
  - Ejemplos de configuración de backend con plantilla de nombre y etiquetas . . . 24
  - Ejemplos de plantillas de nombres . . . 25
  - Puntos a considerar . . . 26
- Compartir un volumen NFS entre espacios de nombres . . . 26
  - Funciones . . . 26
  - Inicio rápido . . . 27
  - Configurar los espacios de nombres de origen y destino . . . 28
  - Eliminar un volumen compartido . . . 29
  - Usar `tridentctl get` para consultar volúmenes subordinados . . . 30
  - Limitaciones . . . 30
  - Para más información . . . 30
- Clonar volúmenes entre espacios de nombres . . . 30
  - Prerrequisitos . . . 30
  - Inicio rápido . . . 31
  - Configurar los espacios de nombres de origen y destino . . . 31
  - Limitaciones . . . 33
- Replicar volúmenes usando SnapMirror . . . 33
  - Requisitos previos de replicación . . . 33
  - Crea un PVC espejado . . . 34
  - Estados de replicación de volumen . . . 37
  - Promover la PVC secundaria durante una conmutación por error no planificada . . . 37
  - Promover la PVC secundaria durante una conmutación por error planificada . . . 38
  - Restablecer una relación de espejo después de una conmutación por error . . . 38
  - Operaciones adicionales . . . 38
  - Actualizar las relaciones de espejo cuando ONTAP esté en línea . . . 39
  - Actualizar las relaciones de réplica cuando ONTAP esté fuera de línea . . . 39
- Utilizar la topología CSI . . . 40

Descripción general . . . 40
Paso 1: Crear un backend que tenga en cuenta la topología . . . 41
Paso 2: Defina las StorageClasses que tengan en cuenta la topología. . . . 43
Paso 3: Crear y usar un PVC . . . 44
Actualizar los backends para incluir `supportedTopologies` . . . 47
Encuentra más información . . . 47
Trabajar con instantáneas . . . 47
Descripción general . . . 47
Cree una instantánea de volumen . . . 48
Cree un PVC a partir de una instantánea de volumen. . . . 49
Importar una instantánea de volumen . . . 50
Recuperar datos de volumen mediante instantáneas . . . 52
Restauración de volumen in situ a partir de una instantánea . . . 52
Eliminar un PV con instantáneas asociadas . . . 54
Implementar un controlador de instantáneas de volumen . . . 54
Enlaces relacionados . . . 55
Trabajar con instantáneas de grupos de volúmenes . . . 55
Crear instantáneas de grupos de volúmenes . . . 56
Recuperar datos de volumen mediante una instantánea de grupo . . . 57
Restauración de volumen in situ a partir de una instantánea . . . 58
Eliminar un PV con instantáneas de grupo asociadas . . . 58
Implementar un controlador de instantáneas de volumen . . . 58
Enlaces relacionados . . . 59

Provisión y gestión de volúmenes

Provisión de un volumen

Cree un PersistentVolumeClaim (PVC) que utilice la StorageClass de Kubernetes configurada para solicitar acceso al PV. Luego puedes montar el panel fotovoltaico en un soporte.

Descripción general

A "*Reclamación de volumen persistente*" (PVC) es una solicitud de acceso al PersistentVolume en el clúster.

El PVC se puede configurar para solicitar el almacenamiento de un tamaño o modo de acceso determinado. Mediante la StorageClass asociada, el administrador del clúster puede controlar más que el tamaño del PersistentVolume y el modo de acceso, como el rendimiento o el nivel de servicio.

Una vez creado el PVC, puede montar el volumen en un soporte.

Crea el PVC

Pasos

1. Crear el PVC.

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

2. Verifique el estado del PVC.

```
kubectl get pvc
```

```
NAME         STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS AGE
pvc-storage  Bound    pv-name    1Gi        RWO                    5m
```

1. Monte el volumen en una cápsula.

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

> Puedes supervisar el progreso usando `kubectl get pod --watch`.

2. Verifique 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 Pod dejará de existir, pero el volumen se mantendrá.

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

Ejemplos de manifiestos

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 a 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 manifiesto de Pod

Estos ejemplos muestran configuraciones básicas para unir el PVC a un módulo.

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 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
```

Referirse a"Objetos de Kubernetes y Trident" Para obtener detalles sobre cómo interactúan las clases de almacenamiento con `PersistentVolumeClaim` y parámetros para controlar cómo Trident gestiona los volúmenes.

Ampliar volúmenes

Trident brinda a los usuarios de Kubernetes la capacidad de expandir sus volúmenes después de su creación. Encuentre información sobre las configuraciones necesarias para expandir volúmenes iSCSI, NFS, SMB, NVMe/TCP y FC.

Expandir un volumen iSCSI

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

> La expansión de volumen iSCSI es compatible con `ontap-san` , `ontap-san-economy` , `solidfire-san` controladores y requiere Kubernetes 1.16 y posterior.

Paso 1: Configure la StorageClass para que admita la expansión de volumen.

Edite la definición de StorageClass para configurar `allowVolumeExpansion` campo a `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 una StorageClass ya existente, edítela para incluir la `allowVolumeExpansion` parámetro.

Paso 2: Cree un PVC con la StorageClass que creó.

Edite la definición de PVC y actualice la `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 esta Reclamación de Volumen Persistente (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: Defina un módulo que conecte el PVC.

Conecte el PV a una cápsula para poder cambiar su tamaño. Existen dos escenarios al redimensionar 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 cambiar el tamaño de un PV no conectado, Trident expande el volumen en el backend de almacenamiento. Una vez que el PVC se vincula a un pod, Trident vuelve a escanear el dispositivo y redimensiona el sistema de archivos. Kubernetes actualiza entonces el tamaño del PVC una vez que la operación de expansión se ha completado correctamente.

En este ejemplo, se crea un pod que utiliza 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: Ampliar el PV

Para cambiar el tamaño del PV creado de 1Gi a 2Gi, edite la definición del PVC y actualícela. `spec.resources.requests.storage` a 2Gi.

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

Paso 5: Validar la expansión

Puedes comprobar que la expansión ha funcionado correctamente verificando el tamaño del PVC, el PV y el volumen del 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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Ampliar un volumen FC

Puede expandir un volumen persistente FC (PV) utilizando el aprovisionador CSI.

> La expansión del volumen FC está soportada por la `ontap-san` controlador y requiere Kubernetes 1.16 y posterior.

Paso 1: Configure la StorageClass para que admita la expansión de volumen.

Edite la definición de StorageClass para configurar `allowVolumeExpansion` campo a `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 una StorageClass ya existente, edítela para incluir la `allowVolumeExpansion` parámetro.

Paso 2: Cree un PVC con la StorageClass que creó.

Edite la definición de PVC y actualice la `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 esta Reclamación de Volumen Persistente (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: Defina un módulo que conecte el PVC.

Conecte el PV a una cápsula para poder cambiar su tamaño. Existen dos escenarios al redimensionar 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 cambiar el tamaño de un PV no conectado, Trident expande el volumen en el backend de almacenamiento. Una vez que el PVC se vincula a un pod, Trident vuelve a escanear el dispositivo y redimensiona el sistema de archivos. Kubernetes actualiza entonces el tamaño del PVC una vez que la

operación de expansión se ha completado correctamente.

En este ejemplo, se crea un pod que utiliza 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: Ampliar el PV

Para cambiar el tamaño del PV creado de 1Gi a 2Gi, edite la definición del PVC y actualícela. `spec.resources.requests.storage` a 2Gi.

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

Paso 5: Validar la expansión

Puedes comprobar que la expansión ha funcionado correctamente verificando el tamaño del PVC, el PV y el volumen del 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 PV NFS aprovisionados en `ontap-nas` , `ontap-nas-economy` , `ontap-nas-flexgroup` , `gcp-cvs` , y `azure-netapp-files` backends.

Paso 1: Configure la StorageClass para que admita la expansión de volumen.

Para cambiar el tamaño de un volumen físico NFS, el administrador primero debe configurar la clase de almacenamiento para permitir la expansión del volumen configurando la `allowVolumeExpansion` campo a `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 ha creado una clase de almacenamiento sin esta opción, simplemente puede editar la clase de almacenamiento existente mediante el uso de `kubectl edit storageclass` para permitir la expansión de volumen.

Paso 2: Cree un PVC con la StorageClass que creó.

```
cat pvc-ontapnas.yaml
```

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

Trident debería crear un PV NFS 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: Ampliar el PV

Para cambiar el tamaño del PV recién creado de 20 MiB a 1 GiB, edite el PVC y configure `spec.resources.requests.storage` hasta 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: Validar la expansión

Puede comprobar que el cambio de tamaño se ha realizado correctamente verificando el tamaño del PVC, el PV y el volumen de Trident :

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

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

volúmenes de importación

Puedes importar volúmenes de almacenamiento existentes como un PV de Kubernetes usando `tridentctl import`.

Descripción general y consideraciones

Podrías importar un volumen a Trident para:

- Contenieriza una aplicación y reutiliza su conjunto de datos existente.
- Utilice un clon de un conjunto de datos para una aplicación efímera.
- Reconstruir un clúster de Kubernetes que ha fallado
- Migrar los datos de la aplicación durante la recuperación ante desastres

Consideraciones

Antes de importar un volumen, revise las siguientes consideraciones.

- Trident solo puede importar volúmenes ONTAP de tipo RW (lectura-escritura). Los volúmenes de tipo DP (protección de datos) son volúmenes de destino SnapMirror . Debes romper la relación de espejo antes de

importar el volumen a Trident.

- Recomendamos importar volúmenes sin conexiones activas. Para importar un volumen que se esté utilizando activamente, clone el volumen y luego realice la importación.

> Esto es especialmente importante para los volúmenes en bloque, ya que Kubernetes desconocería la conexión anterior y podría fácilmente adjuntar un volumen activo a un pod. Esto puede provocar corrupción de datos.

- Aunque `StorageClass` Debe especificarse en un PVC; Trident no utiliza este parámetro durante la importación. Las clases de almacenamiento se utilizan durante la creación de volúmenes para seleccionar entre los grupos disponibles en función de las características de almacenamiento. Dado que el volumen ya existe, no es necesario seleccionar ningún pool durante la importación. Por lo tanto, la importación no fallará incluso si el volumen existe en un backend o pool que no coincide con la clase de almacenamiento especificada en el PVC.
- El volumen existente se determina y se establece en el PVC. Una vez que el controlador de almacenamiento importa el volumen, se crea el PV con una ClaimRef al PVC.
  - La política de reclamaciones está inicialmente establecida para `retain` en el PV. Una vez que Kubernetes enlaza correctamente el PVC y el PV, la política de recuperación se actualiza para que coincida con la política de recuperación de la clase de almacenamiento.
  - Si la política de recuperación de la Clase de Almacenamiento es `delete` El volumen de almacenamiento se eliminará cuando se elimine el PV.
- Por defecto, Trident gestiona el PVC y cambia el nombre del FlexVol volume y del LUN en el backend. Puedes pasar el `--no-manage` bandera para importar un volumen no administrado. Si utilizas `--no -manage` Trident no realiza ninguna operación adicional en el PVC o 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 la clonación y el cambio de tamaño del volumen también se ignoran.

> Esta opción resulta útil si desea utilizar Kubernetes para cargas de trabajo en contenedores, pero por lo demás desea 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 si el volumen fue importado y si el PVC y el PV están gestionados. Esta anotación no debe modificarse ni eliminarse.

Importar un volumen

Puedes utilizar `tridentctl import` para importar un volumen.

Pasos

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

No incluya parámetros adicionales como el nombre del PV o el tamaño del volumen. Esto puede provocar que falle el comando de importación.

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

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

Ejemplos

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

ONTAP NAS y ONTAP NAS FlexGroup

Trident admite la importación de volúmenes mediante el `ontap-nas` y `ontap-nas-flexgroup` conductores.

- Trident no admite la importación de volumen mediante el `ontap-nas-economy` conductor.
- El `ontap-nas` y `ontap-nas-flexgroup` Los controladores no permiten nombres de volumen duplicados.

Cada volumen creado con el `ontap-nas` El controlador es un FlexVol volume en el clúster ONTAP . Importación de volúmenes FlexVol con el `ontap-nas` El controlador funciona igual. Un volumen FlexVol que ya existe en un clúster ONTAP se puede importar como un `ontap-nas` CLORURO DE POLIVINILO. De forma similar, los volúmenes de FlexGroup se pueden importar como `ontap-nas-flexgroup` PVC.

Ejemplos de ONTAP NAS

A continuación se muestra un ejemplo de importación de un volumen administrado y un volumen no administrado.

Volumen gestionado

El siguiente ejemplo importa un volumen llamado `managed_volume` en un backend llamado `ontap_nas` :

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

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

Volumen no gestionado

Al usar el `--no-manage` argumento, Trident no cambia el nombre del volumen.

El siguiente ejemplo importa `unmanaged_volume` en el `ontap_nas` backend:

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-
file> --no-manage

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

ONTAP SAN

Trident admite la importación de volumen mediante el `ontap-san` (iSCSI, NVMe/TCP y FC) y `ontap-san-economy` Conductores.

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

`economy` volúmenes que contienen múltiples LUN.

Ejemplos de ONTAP SAN

A continuación se muestra un ejemplo de importación de un volumen administrado y un volumen no administrado.

Volumen gestionado

Para los volúmenes administrados, Trident cambia el nombre del FlexVol volume a `pvc-<uuid>` formato y el LUN dentro del FlexVol volume a `lun0` .

El siguiente ejemplo importa el `ontap-san-managed` FlexVol volume 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    |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
```

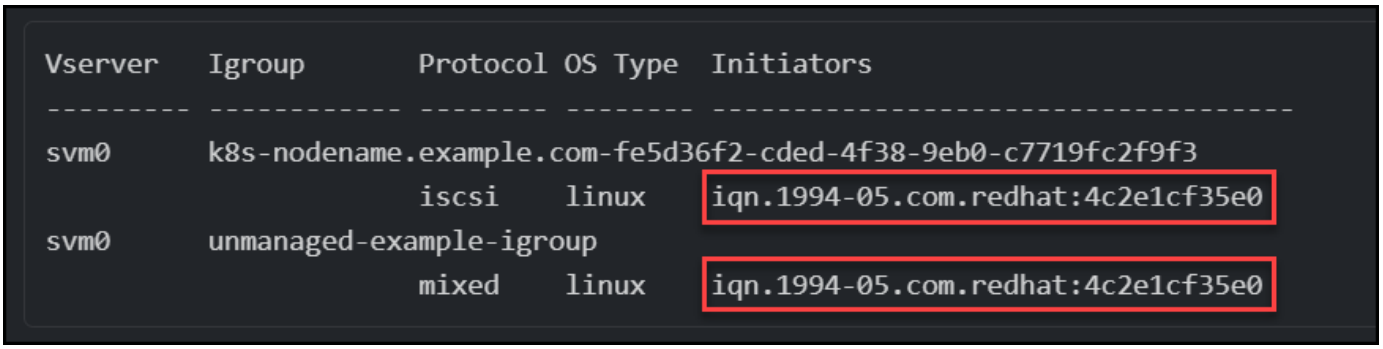
Volumen no gestionado

El siguiente ejemplo importa `unmanaged_example_volume` en el `ontap_san` backend:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
```

Si tiene LUNS asignados a igroups que comparten un IQN con un IQN de nodo de Kubernetes, como se

muestra en el siguiente ejemplo, recibirá el error: `LUN already mapped to initiator(s) in this group` . Deberá eliminar el iniciador o desasignar el LUN para importar el volumen.



Elemento

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

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

Ejemplo de elemento

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

Plataforma de Google Cloud

Trident admite la importación de volúmenes mediante el `gcp-cvs` conductor.

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

Ejemplo de Google Cloud Platform

El siguiente ejemplo importa un `gcp-cvs` volumen en el backend `gcpcvs_YEppr` con la ruta de volumen de `adroit-jolly-swift` .

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

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

Azure NetApp Files

Trident admite la importación de volúmenes mediante el `azure-netapp-files` conductor.

Para importar un volumen de Azure NetApp Files , identifique el volumen por su ruta de acceso. La ruta del volumen es la porción de la ruta de exportación del volumen después de `:/` . Por ejemplo, si la ruta de montaje es `10.0.0.2:/importvol1` , la ruta del volumen es `importvol1` .

Ejemplo de Azure NetApp Files

El siguiente ejemplo importa un `azure-netapp-files` volumen en el backend `azurenetappfiles_40517` con la ruta del volumen `importvol1` .

```
tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

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

Google Cloud NetApp Volumes

Trident admite la importación de volúmenes mediante el `google-cloud-netapp-volumes` conductor.

Ejemplo de Google Cloud NetApp Volumes

El siguiente ejemplo importa un `google-cloud-netapp-volumes` volumen en el backend `backend-tbc-gcnv1` con el volumen `testvoleasiaeast1` .

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

El siguiente ejemplo importa un `google-cloud-netapp-volumes` volumen cuando hay dos volúmenes presentes en la misma región:

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

Personaliza los nombres y etiquetas de los volúmenes.

Con Trident, puedes asignar nombres y etiquetas descriptivos a los volúmenes que crees. Esto te ayuda a identificar y asignar fácilmente los volúmenes a sus respectivos recursos de Kubernetes (PVC). También puede definir plantillas a nivel de backend para crear nombres de volumen personalizados y etiquetas personalizadas; cualquier volumen que cree, importe o clone se ajustará a las plantillas.

Antes de empezar

Compatibilidad con nombres y etiquetas de volumen personalizables:

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

Limitaciones

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

Comportamientos clave de los nombres de volumen personalizables

1. Si se produce un fallo debido a una sintaxis no válida en una plantilla de nombre, falla la creación del backend. Sin embargo, si falla la aplicación de la plantilla, el volumen se nombrará de acuerdo con la convención de nomenclatura existente.

2. El prefijo de almacenamiento no es aplicable cuando un volumen se nombra utilizando una plantilla de nombre de la configuración del backend. Se puede añadir directamente a la plantilla cualquier valor de prefijo deseado.

Ejemplos de configuración de backend con plantilla de nombre y etiquetas

Se pueden definir plantillas de nombres personalizadas a nivel de raíz y/o de grupo.

Ejemplo de nivel raíz

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

Ejemplo de nivel de piscina

```
{
  "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 importaciones de volúmenes, las etiquetas se actualizan solo si el volumen existente tiene etiquetas en un formato específico. Por ejemplo: `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. En el caso de importaciones de volúmenes gestionados, el nombre del volumen sigue la plantilla de nombre definida en el nivel raíz de la definición del backend.
3. Trident no admite el uso de un operador de segmentación con el prefijo de almacenamiento.
4. Si las plantillas no generan nombres de volumen únicos, Trident añadirá 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 excede el límite de nombres, el proceso de creación del volumen falla.

Compartir un volumen NFS entre espacios de nombres

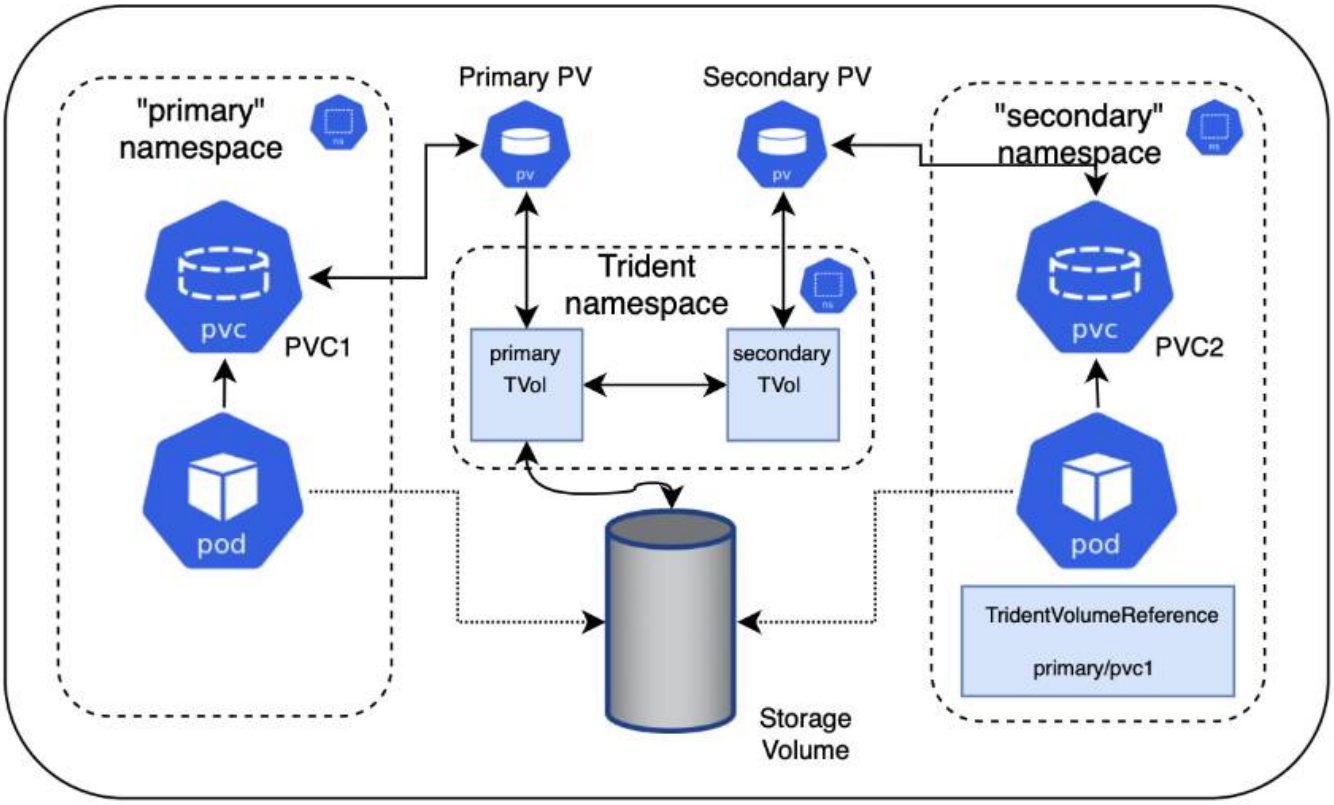
Con Trident, puedes crear un volumen en un espacio de nombres primario y compartirlo en uno o más espacios de nombres secundarios.

Funciones

El CR TridentVolumeReference le permite compartir de forma segura volúmenes NFS ReadWriteMany (RWX) a través de uno o más espacios de nombres de Kubernetes. Esta solución nativa de Kubernetes tiene las siguientes ventajas:

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

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



Inicio rápido

Puedes configurar el uso compartido de volúmenes NFS en tan solo unos pocos pasos.

1 **Configure 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 **Conceder permiso para crear un 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 **Cree una referencia 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 **Cree el PVC subordinado en el espacio de nombres de destino.**

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

Configurar los espacios de nombres de origen y destino

Para garantizar la seguridad, el uso compartido entre espacios de nombres requiere la colaboración y la acción del propietario del espacio de nombres de origen, el administrador 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:** Crear el PVC(`pvc1`) en el espacio de nombres de origen que otorga permiso para compartir con el espacio de nombres de destino(`namespace2`) usando el `shareToNamespace` anotación.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

   Trident crea el PV y su volumen de almacenamiento NFS de backend.

   - Puedes compartir el PVC con múltiples espacios de nombres utilizando una lista delimitada por comas. Por ejemplo, `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4` .
   - Puedes compartir con todos los espacios de nombres usando `*` . Por ejemplo, `trident.netapp.io/shareToNamespace: *`
   - Puedes actualizar el PVC para incluir el `shareToNamespace` anotaciones en cualquier momento.

2. **Administrador del clúster:** Asegúrese 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.
3. **Propietario del espacio de nombres de destino:** Cree un CR TridentVolumeReference en el espacio de nombres de destino que haga 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:** Crear un PVC(`pvc2`) en el espacio de nombres de destino(`namespace2`) usando el `shareFromPVC` Anotación para designar el PVC de origen.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

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

Resultados

Trident lee el `shareFromPVC` Se realiza una anotación en el PVC de destino y se crea el PV de destino como un volumen subordinado sin recursos de almacenamiento propios que apunta al PV de origen y comparte los recursos de almacenamiento del PV de origen. Los valores de destino PVC y PV aparecen vinculados con normalidad.

Eliminar un volumen compartido

Puedes eliminar un volumen que se comparte entre varios espacios de nombres. Trident eliminará el acceso al volumen en el espacio de nombres de origen y mantendrá el acceso para otros espacios de nombres que compartan el volumen. Cuando se eliminan todos los espacios de nombres que hacen referencia al volumen, Trident borra el volumen.

Usar `tridentctl get` para consultar volúmenes subordinados

Utilizando el[`tridentctl` utilidad, puedes ejecutar la `get` comando para obtener volúmenes subordinados. Para obtener más información, consulte el siguiente enlace: ../trident-reference/tridentctl.html[`tridentctl` comandos y opciones].

```
Usage:
  tridentctl get [option]
```

Banderas:

- ``-h, --help`Ayuda para volúmenes.
- `--parentOfSubordinate string`Limitar la consulta al volumen de origen subordinado.
- `--subordinateOf string`Limitar la consulta a los subordinados del volumen.

Limitaciones

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

Para más información

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

- Visita"Compartir volúmenes entre espacios de nombres: Descubra el acceso a volúmenes entre espacios de nombres." .
- Vea la demostración en"NetAppTV" .

Clonar volúmenes entre espacios de nombres

Con Trident, puedes crear nuevos volúmenes utilizando volúmenes existentes o instantáneas de volúmenes de un espacio de nombres diferente dentro del mismo clúster de Kubernetes.

Prerrequisitos

Antes de clonar volúmenes, asegúrese 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 `ontap-san` y `ontap-nas` Controladores de almacenamiento. No se admiten clones de solo lectura.

Inicio rápido

Puedes configurar la clonación de volúmenes en tan solo unos pasos.

Configure 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.

Conceder permiso para crear un 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.

Cree una referencia 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.

Cree el PVC clonado en el espacio de nombres de destino.

El propietario del espacio de nombres de destino crea un PVC para clonar el PVC del espacio de nombres de origen.

Configurar los espacios de nombres de origen y destino

Para garantizar la seguridad, la clonación de volúmenes entre espacios de nombres requiere la colaboración y la 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:** Crear 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 el `cloneToNamespace` anotación.

```
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 múltiples espacios de nombres utilizando una lista delimitada por comas. Por ejemplo, `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4` .
- Puedes compartir con todos los espacios de nombres usando `*` . Por ejemplo, `trident.netapp.io/cloneToNamespace: *`
- Puedes actualizar el PVC para incluir el `cloneToNamespace` anotaciones en cualquier momento.

2. **Administrador del clúster:** Asegúrese de que el control de acceso basado en roles (RBAC) esté configurado correctamente para otorgar permiso al propietario del espacio de nombres de destino para crear el recurso compartido TridentVolumeReference en el espacio de nombres de destino.(`namespace2`).
3. **Propietario del espacio de nombres de destino:** Cree un CR TridentVolumeReference en el espacio de nombres de destino que haga 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:** Crear un PVC(`pvc2`) en el espacio de nombres de destino(`namespace2`) usando el `cloneFromPVC` o `cloneFromSnapshot` , y `cloneFromNamespace` Anotaciones para designar el PVC de origen.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Limitaciones

- Para los PVC aprovisionados mediante controladores ontap-nas-economy, no se admiten clones de solo lectura.

Replicar 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 la recuperación ante desastres. Puede utilizar una definición de recurso personalizado (CRD) con espacio de nombres, denominada relación de espejo de Trident (TMR), para realizar las siguientes operaciones:

- Crear relaciones de simetría entre volúmenes (PVC)
- Eliminar las relaciones de simetría entre volúmenes
- Rompe las relaciones espejo
- Promover el volumen secundario durante situaciones de desastre (conmutaciones por error).
- Realizar una transición sin pérdidas de aplicaciones de un clúster a otro (durante conmutaciones por error o migraciones planificadas).

Requisitos previos de replicación

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

Clústeres ONTAP

- * Trident*: Debe existir la versión 22.10 o posterior de Trident tanto en los clústeres de Kubernetes de origen como de destino que utilizan ONTAP como backend.
- **Licencias**: Las licencias asíncronas de ONTAP SnapMirror que utilizan el paquete de protección de datos deben estar habilitadas tanto en el clúster ONTAP de origen como en el de destino. Referirse a

"Información general sobre 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 de NetApp (NLF), que es un único archivo que habilita múltiples funciones. Referirse a"Licencias incluidas con ONTAP One" Para más información.

> Solo se admite la protección asíncrona de SnapMirror .

Mirando

- **Clúster y SVM**: Los backends de almacenamiento ONTAP deben estar interconectados. Referirse a "Descripción general del emparejamiento de clústeres y SVM" Para más información.

> Asegúrese de que los nombres de SVM utilizados 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 mediante clases de almacenamiento compatibles con los siguientes controladores: **`ontap-nas` : NFS** `ontap-san` : iSCSI **`ontap-san` : FC** `ontap-san` : NVMe/TCP (requiere como mínimo la versión 9.15.1 de ONTAP)

> La replicación de volúmenes mediante SnapMirror no es compatible con los sistemas ASA r2. Para obtener información sobre los sistemas ASA r2, consulte"Conozca los sistemas de almacenamiento ASA r2" .

Crea un PVC espejado

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

Pasos

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

Ejemplo

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "nfs"
  trident.netapp.io/replication: "true"
```

b. Cree un PVC con la StorageClass creada previamente.

Ejemplo

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

c. Cree un CR MirrorRelationship con información local.

Ejemplo

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Trident recupera la información interna del volumen y el estado actual de protección de datos (DP) del volumen, y luego completa el campo de estado de MirrorRelationship.

d. Obtenga el CR TridentMirrorRelationship para obtener el nombre interno y el SVM del PVC.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. Realice los siguientes pasos en el clúster secundario de Kubernetes:
   a. Cree una StorageClass con el parámetro trident.netapp.io/replication: true.

      Ejemplo

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/replication: true
```

   b. Cree un CR MirrorRelationship con información de destino y origen.

      Ejemplo

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Trident creará una relación SnapMirror con el nombre de política de relación configurado (o el predeterminado para ONTAP) y la inicializará.

c. Cree un PVC con la StorageClass creada previamente para que actúe como destino secundario (destino SnapMirror).

Ejemplo

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

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

Estados de replicación de volumen

Una relación de espejo Trident (TMR) es un CRD que representa un extremo de una relación de replicación entre PVC. El TMR de destino tiene un estado, que le indica a Trident cuál es el estado deseado. El TMR de destino tiene los siguientes estados:

- **Establecido**: el PVC local es el volumen de destino de una relación de espejo, y esta es una nueva relación.
- **Promocionado**: el PVC local es de lectura/escritura y se puede montar, sin ninguna relación de espejo actualmente en vigor.
- **Restablecido**: el PVC local es el volumen de destino de una relación de espejo y también estuvo previamente en esa relación de espejo.
  - Debe utilizarse el estado restablecido si el volumen de destino alguna vez estuvo relacionado con el volumen de origen, ya que sobrescribe el contenido del volumen de destino.
  - El estado restablecido fallará si el volumen no estaba previamente relacionado con la fuente.

Promover la PVC secundaria durante una conmutación por error no planificada

Realice el siguiente paso en el clúster de Kubernetes secundario:

- Actualizar el campo *spec.state* de TridentMirrorRelationship a `promoted` .

Promover la PVC secundaria durante una conmutación por error planificada

Durante una conmutación por error planificada (migración), realice los siguientes pasos para promover el PVC secundario:

Pasos

1. En el clúster principal de Kubernetes, cree una instantánea del PVC y espere hasta que se cree la instantánea.
2. En el clúster principal de Kubernetes, cree el CR SnapshotInfo para obtener detalles internos.

   Ejemplo

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

3. En el clúster secundario de Kubernetes, actualice el campo *spec.state* del CR *TridentMirrorRelationship* a *promoted* y *spec.promotedSnapshotHandle* para que sea el internalName de la instantánea.
4. En el clúster secundario de Kubernetes, confirme que el estado (campo status.state) de TridentMirrorRelationship sea promovido.

Restablecer una relación de espejo después de una conmutación por error

Antes de restablecer una relación de espejo, elige el lado que quieres convertir en el nuevo principal.

Pasos

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

Operaciones adicionales

Trident admite las siguientes operaciones en los volúmenes primario y secundario:

Replicar el PVC primario a un nuevo PVC secundario

Asegúrese de que ya dispone de un tubo de PVC primario y un tubo de PVC secundario.

Pasos

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

Cambiar el tamaño de un PVC reflejado, primario o secundario

El tamaño del PVC se puede cambiar como de costumbre; 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, realice una de las siguientes operaciones en el volumen secundario actual:

- Elimine la relación MirrorRelationship en el PVC secundario. Esto rompe la relación de replicación.
- O bien, actualice el campo spec.state a *promoted*.

Eliminar un PVC (que previamente se había duplicado)

Trident comprueba si existen PVC replicados y libera la relación de replicación antes de intentar eliminar el volumen.

Eliminar un TMR

Eliminar un TMR en un lado de una relación reflejada provoca que el TMR restante pase al estado *promocionado* antes de que Trident complete la eliminación. Si el TMR seleccionado para su eliminación ya se encuentra en estado *promocionado*, no existe ninguna relación de réplica y el TMR será eliminado y Trident promoverá el PVC local a *Lectura/Escritura*. Esta eliminación libera los metadatos de SnapMirror para el volumen local en ONTAP. Si este volumen se utiliza en una relación de espejo en el futuro, deberá utilizar un nuevo TMR con un estado de replicación de volumen *establecido* al crear la nueva relación de espejo.

Actualizar las relaciones de espejo cuando ONTAP esté en línea

Las relaciones de espejo se pueden actualizar en cualquier momento después de que se hayan establecido. Puedes utilizar el `state: promoted` o `state: reestablished` campos para actualizar las relaciones. Al promover un volumen de destino a un volumen ReadWrite normal, puede usar *promotedSnapshotHandle* para especificar una instantánea específica a la que restaurar el volumen actual.

Actualizar las relaciones de réplica cuando ONTAP esté fuera de línea

Puede utilizar un CRD para realizar una actualización de SnapMirror sin que Trident tenga conectividad directa con el clúster ONTAP . Consulte el siguiente ejemplo de formato de TridentActionMirrorUpdate:

Ejemplo

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state`refleja el estado del CRD TridentActionMirrorUpdate. Puede tomar un valor de *Succeeded*, *In Progress* o *Failed*.

Utilizar 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

Al utilizar la función de topología CSI, el acceso a los volúmenes se puede limitar a un subconjunto de nodos, según las regiones y las zonas de disponibilidad. Actualmente, los proveedores de servicios en la nube permiten a los administradores de Kubernetes crear nodos basados en zonas. Los nodos pueden ubicarse en diferentes zonas de disponibilidad dentro de una región o en varias regiones. Para facilitar el aprovisionamiento de volúmenes para cargas de trabajo en una arquitectura multizona, Trident utiliza la topología CSI.

Obtenga más información sobre la función de topología CSI. "aquí" .

Kubernetes proporciona dos modos únicos de enlace de volúmenes:

- Con `VolumeBindingMode` empezar a `Immediate` Trident crea el volumen sin tener en cuenta la topología. La vinculación de volúmenes y el aprovisionamiento dinámico se gestionan cuando se crea el PVC. Este es el valor predeterminado `VolumeBindingMode` y es adecuado para clústeres que no imponen restricciones de topología. Los volúmenes persistentes se crean sin depender de los requisitos de programación del pod solicitante.
- Con `VolumeBindingMode` empezar a `WaitForFirstConsumer` La creación y el enlace de un volumen persistente para un PVC se retrasan hasta que se programa y se crea un pod que utiliza el PVC. De esta manera, se crean volúmenes para cumplir con las restricciones de programación impuestas por los requisitos de topología.

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

Lo que necesitarás

Para utilizar la topología CSI, necesita lo siguiente:

- Un clúster de Kubernetes que ejecuta 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 indiquen la topología.(`topology.kubernetes.io/region` y `topology.kubernetes.io/zone`). Estas etiquetas **deben estar presentes en los nodos del clúster** antes de instalar Trident para que Trident tenga en cuenta la topología.

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

Paso 1: Crear un backend que tenga en cuenta la topología

Los sistemas de almacenamiento Trident pueden diseñarse para aprovisionar volúmenes de forma selectiva en función de las zonas de disponibilidad. Cada backend puede incluir una opción. `supportedTopologies` bloque que representa una lista de zonas y regiones admitidas. Para las StorageClasses que utilizan dicho backend, solo se creará un volumen si lo solicita una aplicación programada en una zona o región compatible.

Aquí tenéis un ejemplo de definición de backend:

YAML

```yaml
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-a
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-b
```

JSON

```json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-a"
    },
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-b"
    }
  ]
}
```

`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 una StorageClass. Para las 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 cada grupo de almacenamiento también. Vea 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, el `region` y `zone` Las etiquetas indican la ubicación del depósito de almacenamiento. `topology.kubernetes.io/region` y `topology.kubernetes.io/zone` dictar desde dónde se pueden consumir los grupos de almacenamiento.

Paso 2: Defina las StorageClasses que tengan en cuenta la topología.

En función de las etiquetas de topología que se proporcionan a los nodos del 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 utilizar los volúmenes aprovisionados por Trident.

Vea el siguiente ejemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata: null
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions: null
  - key: topology.kubernetes.io/zone
    values:
      - us-east1-a
      - us-east1-b
  - key: topology.kubernetes.io/region
    values:
      - us-east1
parameters:
  fsType: ext4
```

En la definición de StorageClass proporcionada anteriormente, `volumeBindingMode` está configurado para `WaitForFirstConsumer` . Los PVC que se soliciten con esta StorageClass no se procesarán hasta que se haga referencia a ellos en un pod. Y, `allowedTopologies` proporciona las zonas y la región que se utilizarán. El `netapp-san-us-east1` StorageClass crea PVC en el `san-backend-us-east1` Backend definido anteriormente.

Paso 3: Crear y usar un PVC

Una vez creada la StorageClass y asignada a un backend, ya puede crear PVC.

Vea 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
```

La creación de un PVC utilizando este manifiesto daría como resultado lo siguiente:

```
kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

Para que Trident cree un volumen y lo una al PVC, utilice el PVC en una cápsula. Vea el siguiente ejemplo:

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

Esta especificación de pod indica a Kubernetes que programe el pod en los nodos que están presentes en el `us-east1` región, y elige cualquier nodo que esté presente en la `us-east1-a` o `us-east1-b` zonas.

Vea el siguiente resultado:

```
kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

Actualizar los backends para incluir `supportedTopologies`

Los sistemas backend 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 utilizará para los PVC posteriores.

Encuentra más información

- "Gestionar recursos para contenedores"
- "selector de nodo"
- "Afinidad y antiafinidad"
- "Manchas y tolerancias"

Trabajar con instantáneas

Las instantáneas de volúmenes persistentes (PV) de Kubernetes permiten realizar copias puntuales de los volúmenes. Puede crear una instantánea de un volumen creado con Trident, importar una instantánea creada fuera de Trident, crear un nuevo volumen a partir de una instantánea existente y recuperar datos de volumen a partir de instantáneas.

Descripción general

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

Antes de empezar

Para trabajar con instantáneas, debe disponer de un controlador de instantáneas externo y definiciones de recursos personalizados (CRD). Esta es responsabilidad del orquestador de Kubernetes (por ejemplo: Kubeadm, GKE, OpenShift).

Si su distribución de Kubernetes no incluye el controlador de instantáneas ni los CRD, consulteImplementar un controlador de instantáneas de volumen .

No cree un controlador de instantáneas si va a crear instantáneas de volumen bajo demanda en un entorno GKE. GKE utiliza un controlador de instantáneas integrado y oculto.

Cree una instantánea de volumen

Pasos

1. Crear una `VolumeSnapshotClass` Para obtener más información, consulte"Clase de instantánea de volumen" .
   - El `driver` señala al controlador Trident CSI.
   - `deletionPolicy`puede ser `Delete o `Retain . Cuando se configura para `Retain` La instantánea física subyacente en el clúster de almacenamiento se conserva incluso cuando `VolumeSnapshot` El objeto ha sido eliminado.

     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. Cree 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 `VolumeSnapshotContent` objeto para el `pvc1-snap` VolumeSnapshot describiéndolo. El `Snapshot Content Name` identifica el objeto VolumeSnapshotContent que sirve a esta instantánea. El `Ready To Use` El parámetro indica que la instantánea se puede utilizar para crear un nuevo PVC.

```
kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
...
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
...
```

Cree un PVC a partir de una instantánea de volumen.

Puedes utilizar `dataSource` para crear un PVC utilizando un VolumeSnapshot llamado `<pvc-name>` como fuente de los datos. Una vez creado el PVC, se puede acoplar a una cápsula y utilizar como cualquier otro PVC.

> El PVC se creará en el mismo backend que el volumen fuente. Referirse a"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 utilizando `pvc1-snap` como fuente de datos.

```
cat pvc-from-snap.yaml
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Importar una instantánea de volumen

Trident apoya a"Proceso de instantáneas preaprovisionadas de Kubernetes" para permitir que el administrador del clúster cree un `VolumeSnapshotContent` Objetos e instantáneas de importación creadas fuera de Trident.

Antes de empezar

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

Pasos

1. **Administrador del clúster:** Crear un `VolumeSnapshotContent` Objeto que hace referencia a la instantánea del backend. Esto inicia el flujo de trabajo de instantáneas en Trident.
   - Especifique el nombre de la instantánea del backend en `annotations` como `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">` .
   - Especificar `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` en `snapshotHandle` Esta es la única información proporcionada a Trident por el capturador de instantáneas externo en el `ListSnapshots` llamar.

     > El `<volumeSnapshotContentName>` No siempre puede coincidir el nombre de la instantánea del 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. **Administrador del clúster:** Crear el `VolumeSnapshot` CR que hace referencia al `VolumeSnapshotContent` objeto. Esta solicitud permite el uso de `VolumeSnapshot` en un espacio de nombres determinado.

   Ejemplo

   El siguiente ejemplo crea un `VolumeSnapshot` CR llamado `import-snap` que hace referencia a `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 se requiere ninguna acción):** El generador de instantáneas externo reconoce la instantánea recién creada `VolumeSnapshotContent` y ejecuta el `ListSnapshots` llamar. Trident crea el `TridentSnapshot` .
   - El capturador externo establece el `VolumeSnapshotContent` a `readyToUse` y el `VolumeSnapshot` a `true` .
   - El Trident regresa `readyToUse=true` .
4. **Cualquier usuario:** Crea un `PersistentVolumeClaim` para hacer referencia al nuevo `VolumeSnapshot` , donde el `spec.dataSource` (o `spec.dataSourceRef`) el nombre es el

`VolumeSnapshot` nombre.

Ejemplo

El siguiente ejemplo crea un 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
```

Recuperar datos de volumen mediante instantáneas

El directorio de instantáneas está oculto de forma predeterminada para facilitar la máxima compatibilidad de los volúmenes aprovisionados mediante `ontap-nas` y `ontap-nas-economy` conductores. Habilitar el `.snapshot` directorio para recuperar datos directamente desde instantáneas.

Utilice la CLI de ONTAP para restaurar instantáneas de volumen para restaurar un volumen a un estado registrado en una instantánea anterior.

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

> Al restaurar una copia de instantánea, se sobrescribe la configuración de volumen existente. Los cambios realizados en los datos del volumen después de la creación de la copia de instantánea se pierden.

Restauración de volumen in situ a partir de una instantánea

Trident proporciona una restauración de volumen rápida e in situ a partir de una instantánea utilizando la `TridentActionSnapshotRestore` (TASR) CR. Esta solicitud de cambio (CR) funciona como una acción imperativa de Kubernetes y no persiste una vez finalizada la operación.

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

Antes de empezar

Debe tener un PVC vinculado y una instantánea de volumen disponible.

- Verifique que el estado del PVC esté vinculado.

```
kubectl get pvc
```

- Verifique que la instantánea del volumen esté lista para usar.

```
kubectl get vs
```

Pasos

1. Cree el TASR CR. Este ejemplo crea una solicitud de cambio para PVC. `pvc1` y instantánea de volumen `pvc1-snapshot` .

   El CR TASR 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. Aplique el CR para restaurar desde la instantánea. Este ejemplo restaura desde una instantánea. `pvc1` .

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

Resultados

Trident restaura los datos a partir de la instantánea. Puedes verificar el estado de 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. Deberá repetir la operación.
- Los usuarios de Kubernetes sin acceso de administrador podrían necesitar que el administrador les otorgue permiso para crear un CR TASR 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 de "Eliminación". Elimine las instantáneas de volumen para borrar el volumen de Trident .

Implementar un controlador de instantáneas de volumen

Si tu distribución de Kubernetes no incluye el controlador de instantáneas y los CRD, puedes implementarlos de la siguiente manera.

Pasos

1. Crear CRD de instantáneas de volumen.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Crea el controlador de instantáneas.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

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

Enlaces relacionados

- "Instantáneas de volumen"
- "Clase de instantánea de volumen"

Trabajar con instantáneas de grupos de volúmenes

Instantáneas de grupos de volúmenes de Kubernetes de volúmenes persistentes (PV) NetApp Trident ofrece la capacidad de crear instantáneas de múltiples volúmenes (un grupo de instantáneas de volumen). Esta instantánea del grupo de volúmenes representa copias de múltiples volúmenes tomadas en el mismo momento.

VolumeGroupSnapshot es una función beta en Kubernetes con API beta. Kubernetes 1.32 es la versión mínima requerida para VolumeGroupSnapshot.

Crear instantáneas de grupos de volúmenes

La instantánea de grupo de volúmenes es compatible con `ontap-san` Controlador, solo para el protocolo iSCSI, aún no compatible con Fibre Channel (FCP) ni NVMe/TCP. Antes de comenzar

- Asegúrese de que su versión de Kubernetes sea K8s 1.32 o superior.
- Para trabajar con instantáneas, debe disponer de un controlador de instantáneas externo y definiciones de recursos personalizados (CRD). Esta es responsabilidad del orquestador de Kubernetes (por ejemplo: Kubeadm, GKE, OpenShift).

  Si su distribución de Kubernetes no incluye el controlador de instantáneas externo ni los CRD, consulteImplementar un controlador de instantáneas de volumen .

  No cree un controlador de instantáneas si va a crear instantáneas de grupos de volúmenes bajo demanda en un entorno GKE. GKE utiliza un controlador de instantáneas integrado y oculto.

- En el archivo YAML del controlador de instantáneas, configure el `CSIVolumeGroupSnapshot` Establezca el valor de la función en 'true' para garantizar que la instantánea del grupo de volúmenes esté habilitada.
- Cree las clases de instantáneas de grupo de volúmenes necesarias antes de crear una instantánea de grupo de volúmenes.
- Asegúrese de que todos los PVC/volúmenes estén en el mismo SVM para poder crear VolumeGroupSnapshot.

Pasos

- Cree una clase VolumeGroupSnapshotClass antes de crear un objeto VolumeGroupSnapshot. Para obtener más información, consulte"Clase de instantánea de grupo de volumen" .

```
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
```

- Cree PVC con las etiquetas necesarias utilizando las clases de almacenamiento existentes, o agregue estas etiquetas a los PVC existentes.

  El siguiente ejemplo crea el PVC utilizando `pvc1-group-snap` como fuente de datos y etiqueta `consistentGroupSnapshot: groupA` . Define la clave y el valor de la etiqueta según tus requisitos.

```
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`) especificado 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
```

Recuperar datos de volumen mediante una instantánea de grupo

Puede restaurar volúmenes persistentes individuales utilizando las instantáneas individuales que se han creado como parte de la instantánea del grupo de volúmenes. No se puede recuperar la instantánea del grupo de volúmenes como una unidad.

Utilice la CLI de ONTAP para restaurar instantáneas de volumen para restaurar un volumen a un estado registrado en una instantánea anterior.

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

Al restaurar una copia de instantánea, se sobrescribe la configuración de volumen existente. Los cambios realizados en los datos del volumen después de la creación de la copia de instantánea se pierden.

Restauración de volumen in situ a partir de una instantánea

Trident proporciona una restauración de volumen rápida e in situ a partir de una instantánea utilizando la `TridentActionSnapshotRestore` (TASR) CR. Esta solicitud de cambio (CR) funciona como una acción imperativa de Kubernetes y no persiste una vez finalizada la operación.

Para obtener más información, consulte "Restauración de volumen in situ a partir de una instantánea".

Eliminar un PV con instantáneas de grupo asociadas

Al eliminar una instantánea de volumen de grupo:

- Puede eliminar VolumeGroupSnapshots en su totalidad, no instantáneas individuales dentro del grupo.
- Si se eliminan volúmenes persistentes mientras existe una instantánea para ese volumen persistente, Trident moverá ese volumen a un estado de "eliminación" porque la instantánea debe eliminarse antes de que el volumen pueda eliminarse de forma segura.
- Si se ha creado un clon utilizando una instantánea agrupada y luego se va a eliminar el grupo, se iniciará una operación de división en clon y el grupo no se podrá eliminar hasta que se complete la división.

Implementar un controlador de instantáneas de volumen

Si tu distribución de Kubernetes no incluye el controlador de instantáneas y los CRD, puedes implementarlos de la siguiente manera.

Pasos

1. Crear CRD de instantáneas de volumen.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. Crea el controlador de instantáneas.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

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

Enlaces relacionados

- "Clase de instantánea de grupo de volumen"
- "Instantáneas de volumen"