Fornire e gestire i volumi

Trident

NetApp
January 15, 2026

Sommario

Fornire e gestire i volumi

Fornire un volume

Creare un PersistentVolumeClaim (PVC) che utilizzi la StorageClass Kubernetes configurata per richiedere l'accesso al PV. È quindi possibile montare il fotovoltaico su un pod.

Panoramica

UN "*PersistentVolumeClaim*" (PVC) è una richiesta di accesso al PersistentVolume sul cluster.

Il PVC può essere configurato per richiedere una determinata dimensione di archiviazione o modalità di accesso. Utilizzando la StorageClass associata, l'amministratore del cluster può controllare molto più della dimensione e della modalità di accesso di PersistentVolume, ad esempio le prestazioni o il livello di servizio.

Dopo aver creato il PVC, è possibile montare il volume in un pod.

Creare il PVC

Passi

1. Creare il PVC.

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

2. Verificare lo stato del PVC.

```
kubectl get pvc
```

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

1. Montare il volume in un pod.

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

> Puoi monitorare i progressi utilizzando `kubectl get pod --watch`.

2. Verificare che il volume sia montato su `/my/mount/path`.

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

3. Ora puoi eliminare il Pod. L'applicazione Pod non esisterà più, ma il volume rimarrà.

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

Esempi di manifesti

Manifesti di esempio PersistentVolumeClaim

Questi esempi mostrano le opzioni di configurazione di base del PVC.

PVC con accesso RWO

Questo esempio mostra un PVC di base con accesso RWO associato a una StorageClass denominata `basic-csi` .

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

PVC con NVMe/TCP

Questo esempio mostra un PVC di base per NVMe/TCP con accesso RWO associato a una StorageClass denominata `protection-gold` .

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

Campioni di manifesti del pod

Questi esempi mostrano le configurazioni di base per fissare il PVC a un pod.

Configurazione di base

```
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
```

Configurazione NVMe/TCP di base

```
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
```

Fare riferimento a"Oggetti Kubernetes e Trident" per i dettagli su come le classi di archiviazione interagiscono con `PersistentVolumeClaim` e parametri per controllare il modo in cui Trident approvvigiona i volumi.

Espandi i volumi

Trident offre agli utenti di Kubernetes la possibilità di espandere i propri volumi dopo averli creati. Trova informazioni sulle configurazioni necessarie per espandere i volumi iSCSI, NFS, SMB, NVMe/TCP e FC.

Espandere un volume iSCSI

È possibile espandere un volume persistente iSCSI (PV) utilizzando il provisioner CSI.

> L'espansione del volume iSCSI è supportata da `ontap-san` , `ontap-san-economy` , `solidfire-san` driver e richiede Kubernetes 1.16 e versioni successive.

Passaggio 1: configurare StorageClass per supportare l'espansione del volume

Modificare la definizione StorageClass per impostare `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
```

Per una StorageClass già esistente, modificarla per includere `allowVolumeExpansion` parametro.

Passaggio 2: creare un PVC con la StorageClass creata

Modifica la definizione PVC e aggiornala `spec.resources.requests.storage` per riflettere la nuova dimensione desiderata, che deve essere maggiore della dimensione originale.

```
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 Persistent Volume (PV) e lo associa a questo Persistent Volume Claim (PVC).

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

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

Fase 3: definire un pod che collega il PVC

Collega il PV a un pod per ridimensionarlo. Esistono due scenari quando si ridimensiona un PV iSCSI:

- Se il PV è collegato a un pod, Trident espande il volume sul backend di archiviazione, esegue una nuova scansione del dispositivo e ridimensiona il file system.
- Quando si tenta di ridimensionare un PV non collegato, Trident espande il volume sul backend di archiviazione. Dopo che il PVC è stato associato a un pod, Trident esegue una nuova scansione del dispositivo e ridimensiona il file system. Kubernetes aggiorna quindi la dimensione del PVC una volta completata correttamente l'operazione di espansione.

In questo esempio, viene creato un pod che utilizza il `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
```

Fase 4: Espandi il PV

Per ridimensionare il PV creato da 1Gi a 2Gi, modificare la definizione PVC e aggiornare `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
 # ...
```

Fase 5: convalidare l'espansione

È possibile verificare che l'espansione abbia funzionato correttamente controllando le dimensioni del PVC, del PV e del volume 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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Espandi un volume FC

È possibile espandere un FC Persistent Volume (PV) utilizzando il provisioner CSI.

> L'espansione del volume FC è supportata da `ontap-san` driver e richiede Kubernetes 1.16 e versioni successive.

Passaggio 1: configurare StorageClass per supportare l'espansione del volume

Modificare la definizione StorageClass per impostare `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
```

Per una StorageClass già esistente, modificarla per includere `allowVolumeExpansion` parametro.

Passaggio 2: creare un PVC con la StorageClass creata

Modifica la definizione PVC e aggiornala `spec.resources.requests.storage` per riflettere la nuova dimensione desiderata, che deve essere maggiore della dimensione originale.

```
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 Persistent Volume (PV) e lo associa a questo Persistent Volume Claim (PVC).

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

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

Fase 3: definire un pod che collega il PVC

Collega il PV a un pod per ridimensionarlo. Esistono due scenari quando si ridimensiona un FC PV:

- Se il PV è collegato a un pod, Trident espande il volume sul backend di archiviazione, esegue una nuova scansione del dispositivo e ridimensiona il file system.
- Quando si tenta di ridimensionare un PV non collegato, Trident espande il volume sul backend di archiviazione. Dopo che il PVC è stato associato a un pod, Trident esegue una nuova scansione del dispositivo e ridimensiona il file system. Kubernetes aggiorna quindi la dimensione del PVC una volta completata correttamente l'operazione di espansione.

In questo esempio, viene creato un pod che utilizza il `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
```

Fase 4: Espandi il PV

Per ridimensionare il PV creato da 1Gi a 2Gi, modificare la definizione PVC e aggiornare `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
 # ...
```

Fase 5: convalidare l'espansione

È possibile verificare che l'espansione abbia funzionato correttamente controllando le dimensioni del PVC, del PV e del volume 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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Espandere un volume NFS

Trident supporta l'espansione del volume per i PV NFS forniti su `ontap-nas` , `ontap-nas-economy` , `ontap-nas-flexgroup` , `gcp-cvs` , E `azure-netapp-files` backend.

Passaggio 1: configurare StorageClass per supportare l'espansione del volume

Per ridimensionare un PV NFS, l'amministratore deve prima configurare la classe di archiviazione per consentire l'espansione del volume impostando `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
```

Se hai già creato una classe di archiviazione senza questa opzione, puoi semplicemente modificare la classe

di archiviazione esistente utilizzando `kubectl edit storageclass` per consentire l'espansione del volume.

Passaggio 2: creare un PVC con la StorageClass creata

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

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

Trident dovrebbe creare un PV NFS da 20 MiB per questo 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
```

Fase 3: Espandi il PV

Per ridimensionare il PV da 20 MiB appena creato a 1 GiB, modificare il PVC e impostare `spec.resources.requests.storage` a 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
# ...
```

Fase 4: convalidare l'espansione

È possibile verificare che il ridimensionamento abbia funzionato correttamente controllando le dimensioni del PVC, del PV e del volume 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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Volumi di importazione

È possibile importare volumi di archiviazione esistenti come PV Kubernetes utilizzando `tridentctl import`.

Panoramica e considerazioni

È possibile importare un volume in Trident per:

- Containerizzare un'applicazione e riutilizzare il suo set di dati esistente
- Utilizzare un clone di un set di dati per un'applicazione effimera
- Ricostruisci un cluster Kubernetes non riuscito
- Migrare i dati dell'applicazione durante il ripristino di emergenza

Considerazioni

Prima di importare un volume, esaminare le seguenti considerazioni.

- Trident può importare solo volumi ONTAP di tipo RW (lettura-scrittura). I volumi di tipo DP (protezione dati) sono volumi di destinazione SnapMirror . Prima di importare il volume in Trident, è necessario interrompere

la relazione mirror.

- Suggeriamo di importare volumi senza connessioni attive. Per importare un volume utilizzato attivamente, clonare il volume e quindi eseguire l'importazione.

> Ciò è particolarmente importante per i volumi a blocchi, poiché Kubernetes non sarebbe a conoscenza della connessione precedente e potrebbe facilmente collegare un volume attivo a un pod. Ciò può causare il danneggiamento dei dati.

- Anche se `StorageClass` deve essere specificato su un PVC, Trident non utilizza questo parametro durante l'importazione. Le classi di archiviazione vengono utilizzate durante la creazione del volume per selezionare tra i pool disponibili in base alle caratteristiche di archiviazione. Poiché il volume esiste già, non è richiesta alcuna selezione del pool durante l'importazione. Pertanto, l'importazione non fallirà anche se il volume esiste su un backend o pool che non corrisponde alla classe di archiviazione specificata nel PVC.
- La dimensione del volume esistente viene determinata e impostata nel PVC. Dopo che il volume è stato importato dal driver di archiviazione, il PV viene creato con un ClaimRef al PVC.
  - La politica di recupero è inizialmente impostata su `retain` nel PV. Dopo che Kubernetes ha associato correttamente PVC e PV, la policy di recupero viene aggiornata per corrispondere alla policy di recupero della classe di archiviazione.
  - Se la politica di recupero della classe di archiviazione è `delete` , il volume di archiviazione verrà eliminato quando il PV verrà eliminato.
- Per impostazione predefinita, Trident gestisce il PVC e rinomina il FlexVol volume e il LUN sul backend. Puoi passare il `--no-manage` flag per importare un volume non gestito. Se usi `--no-manage` , Trident non esegue alcuna operazione aggiuntiva sul PVC o sul PV per l'intero ciclo di vita degli oggetti. Il volume di archiviazione non viene eliminato quando si elimina il PV e anche altre operazioni come la clonazione del volume e il ridimensionamento del volume vengono ignorate.

> Questa opzione è utile se si desidera utilizzare Kubernetes per carichi di lavoro containerizzati ma si desidera gestire il ciclo di vita del volume di archiviazione al di fuori di Kubernetes.

- Viene aggiunta un'annotazione al PVC e al PV che ha il duplice scopo di indicare che il volume è stato importato e se il PVC e il PV sono gestiti. Questa annotazione non deve essere modificata o rimossa.

Importa un volume

Puoi usare `tridentctl import` per importare un volume.

Passi

1. Creare il file Persistent Volume Claim (PVC) (ad esempio, `pvc.yaml`) che verrà utilizzato per creare il PVC. Il file PVC dovrebbe includere `name` , `namespace` , `accessModes` , E `storageClassName` . Facoltativamente, puoi specificare `unixPermissions` nella tua definizione di PVC.

   Di seguito è riportato un esempio di specifica minima:

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

Non includere parametri aggiuntivi come il nome PV o la dimensione del volume. Ciò può causare il fallimento del comando di importazione.

2. Utilizzare il `tridentctl import` comando per specificare il nome del backend Trident contenente il volume e il nome che identifica in modo univoco il volume nello storage (ad esempio: ONTAP FlexVol, Element Volume, percorso Cloud Volumes Service). IL `-f` L'argomento è necessario per specificare il percorso del file PVC.

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

Esempi

Esaminare i seguenti esempi di importazione di volumi per i driver supportati.

ONTAP NAS e ONTAP NAS FlexGroup

Trident supporta l'importazione di volumi utilizzando `ontap-nas` E `ontap-nas-flexgroup` conducenti.

- Trident non supporta l'importazione di volumi utilizzando `ontap-nas-economy` autista.
- IL `ontap-nas` E `ontap-nas-flexgroup` i driver non consentono nomi di volume duplicati.

Ogni volume creato con il `ontap-nas` il driver è un FlexVol volume sul cluster ONTAP . Importazione di volumi FlexVol con `ontap-nas` il driver funziona allo stesso modo. Un volume FlexVol già esistente su un cluster ONTAP può essere importato come `ontap-nas` PVC. Allo stesso modo, i volumi FlexGroup possono essere importati come `ontap-nas-flexgroup` PVC.

Esempi di ONTAP NAS

Di seguito è riportato un esempio di importazione di un volume gestito e di un volume non gestito.

Volume gestito

L'esempio seguente importa un volume denominato `managed_volume` su un backend denominato `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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Volume non gestito

Quando si utilizza il `--no-manage` argomento, Trident non rinomina il volume.

Il seguente esempio importa `unmanaged_volume` sul `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 supporta l'importazione di volumi utilizzando `ontap-san` (iSCSI, NVMe/TCP e FC) e `ontap-san-economy` conducenti.

Trident può importare volumi ONTAP SAN FlexVol che contengono un singolo LUN. Ciò è coerente con il `ontap-san` driver, che crea un FlexVol volume per ogni PVC e un LUN all'interno del FlexVol volume. Trident importa il FlexVol volume e lo associa alla definizione PVC. Trident può importare `ontap-san-economy`

volumi che contengono più LUN.

Esempi ONTAP SAN

Di seguito è riportato un esempio di importazione di un volume gestito e di un volume non gestito.

Volume gestito

Per i volumi gestiti, Trident rinomina il FlexVol volume in `pvc-<uuid>` formato e LUN all'interno del FlexVol volume per `lun0` .

L'esempio seguente importa il `ontap-san-managed` FlexVol volume presente sul `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    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

Volume non gestito

Il seguente esempio importa `unmanaged_example_volume` sul `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   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

Se hai LUNS mappati su igroup che condividono un IQN con un IQN del nodo Kubernetes, come mostrato nell'esempio seguente, riceverai l'errore: `LUN already mapped to initiator(s) in this group` .

Per importare il volume sarà necessario rimuovere l'iniziatore o annullare la mappatura del LUN.

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

Elemento

Trident supporta il software NetApp Element e l'importazione di volumi NetApp HCI utilizzando `solidfire-san` autista.

> Il driver Element supporta nomi di volume duplicati. Tuttavia, Trident restituisce un errore se sono presenti nomi di volume duplicati. Come soluzione alternativa, clonare il volume, fornire un nome di volume univoco e importare il volume clonato.

Esempio di elemento

L'esempio seguente importa un `element-managed` volume sul 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    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

Piattaforma Google Cloud

Trident supporta l'importazione di volumi utilizzando `gcp-cvs` autista.

> Per importare un volume supportato dal servizio NetApp Cloud Volumes Service in Google Cloud Platform, identifica il volume tramite il suo percorso. Il percorso del volume è la parte del percorso di esportazione del volume dopo il `:/` . Ad esempio, se il percorso di esportazione è `10.0.0.1:/adroit-jolly-swift` , il percorso del volume è `adroit-jolly-swift` .

Esempio di Google Cloud Platform

L'esempio seguente importa un `gcp-cvs` volume sul backend `gcpcvs_YEppr` con il percorso del volume di `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 supporta l'importazione di volumi utilizzando `azure-netapp-files` autista.

> Per importare un volume Azure NetApp Files , identifica il volume tramite il suo percorso. Il percorso del volume è la parte del percorso di esportazione del volume dopo il `:/` . Ad esempio, se il percorso di montaggio è `10.0.0.2:/importvol1` , il percorso del volume è `importvol1` .

Esempio Azure NetApp Files

L'esempio seguente importa un `azure-netapp-files` volume sul backend `azurenetappfiles_40517` con il percorso del volume `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 supporta l'importazione di volumi utilizzando `google-cloud-netapp-volumes` autista.

Esempio Google Cloud NetApp Volumes

L'esempio seguente importa un `google-cloud-netapp-volumes` volume sul backend `backend-tbc-gcnv1` con il volume `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
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

L'esempio seguente importa un `google-cloud-netapp-volumes` volume quando due volumi sono presenti nella stessa regione:

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

Personalizza i nomi e le etichette dei volumi

Con Trident puoi assegnare nomi ed etichette significativi ai volumi che crei. Ciò consente di identificare e mappare facilmente i volumi alle rispettive risorse Kubernetes (PVC). È anche possibile definire modelli a livello di backend per creare nomi di volume ed etichette personalizzate; tutti i volumi creati, importati o clonati aderiranno ai modelli.

Prima di iniziare

Supporto per nomi e etichette di volume personalizzabili:

1. Operazioni di creazione, importazione e clonazione del volume.
2. Nel caso del driver ontap-nas-economy, solo il nome del volume Qtree è conforme al modello di nome.
3. Nel caso del driver ontap-san-economy, solo il nome LUN è conforme al modello di nome.

Limitazioni

1. I nomi dei volumi personalizzabili sono compatibili solo con i driver ONTAP on-premises.
2. I nomi dei volumi personalizzabili non si applicano ai volumi esistenti.

Comportamenti chiave dei nomi dei volumi personalizzabili

1. Se si verifica un errore a causa di una sintassi non valida in un modello di nome, la creazione del backend fallisce. Tuttavia, se l'applicazione del modello fallisce, il volume verrà denominato in base alla convenzione di denominazione esistente.
2. Il prefisso di archiviazione non è applicabile quando un volume viene denominato utilizzando un modello di

nome dalla configurazione backend. È possibile aggiungere direttamente al modello qualsiasi valore di prefisso desiderato.

Esempi di configurazione del backend con modello di nome ed etichette

È possibile definire modelli di nomi personalizzati a livello di radice e/o di pool.

Esempio di livello radice

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

Esempio di livello della 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 }}"
      }
    }
  ]
}
```

Esempi di modelli di nomi

Esempio 1:

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

Esempio 2:

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

Punti da considerare

1. Nel caso di importazioni di volumi, le etichette vengono aggiornate solo se il volume esistente ha etichette in un formato specifico. Per esempio: `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. Nel caso di importazioni di volumi gestiti, il nome del volume segue il modello di nome definito a livello radice nella definizione del backend.
3. Trident non supporta l'uso di un operatore slice con il prefisso storage.
4. Se i modelli non generano nomi di volume univoci, Trident aggiungerà alcuni caratteri casuali per creare nomi di volume univoci.
5. Se il nome personalizzato per un volume economico NAS supera i 64 caratteri, Trident nominerà i volumi in base alla convenzione di denominazione esistente. Per tutti gli altri driver ONTAP , se il nome del volume supera il limite, il processo di creazione del volume fallisce.

Condividere un volume NFS tra gli spazi dei nomi

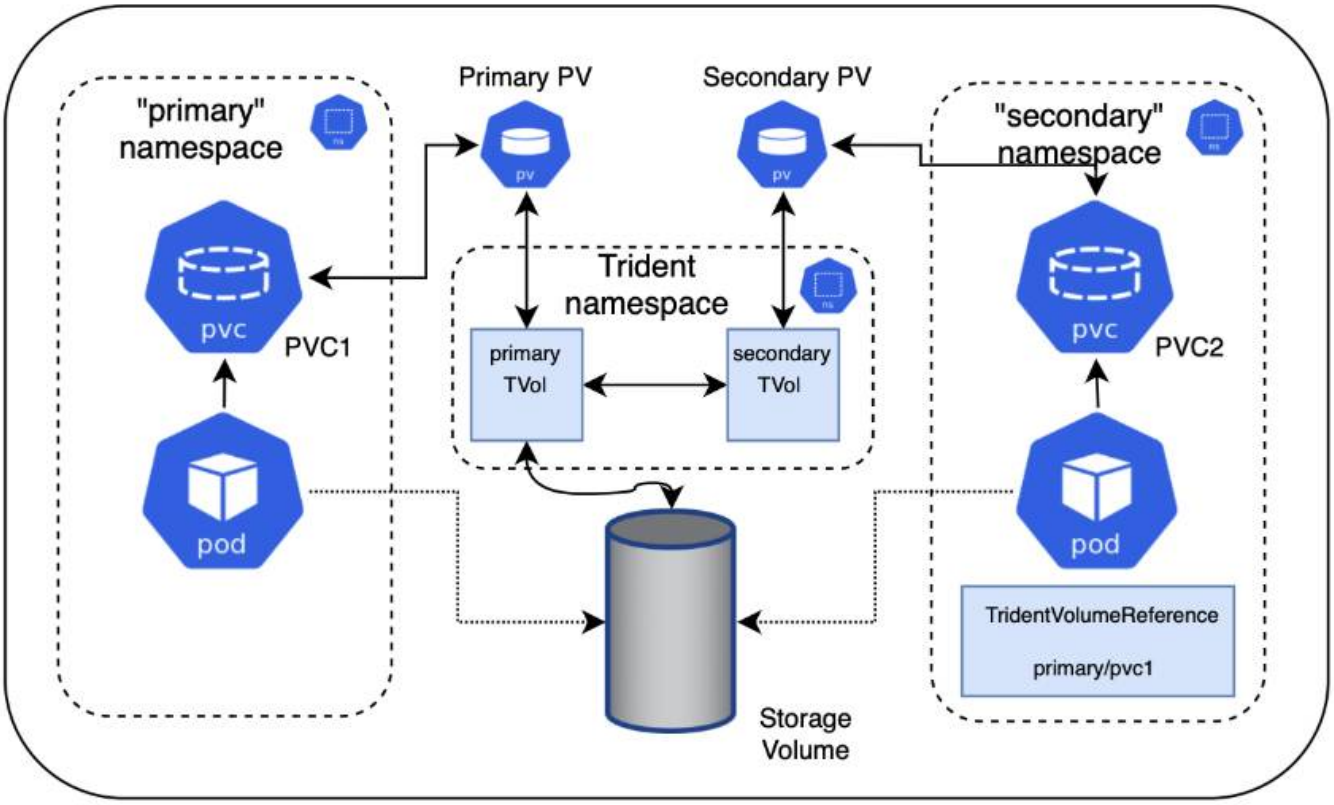
Utilizzando Trident, è possibile creare un volume in uno spazio dei nomi primario e condividerlo in uno o più spazi dei nomi secondari.

Caratteristiche

TridentVolumeReference CR consente di condividere in modo sicuro volumi NFS ReadWriteMany (RWX) su uno o più namespace Kubernetes. Questa soluzione nativa di Kubernetes offre i seguenti vantaggi:

- Più livelli di controllo degli accessi per garantire la sicurezza
- Funziona con tutti i driver di volume Trident NFS
- Nessuna dipendenza da tridentctl o da qualsiasi altra funzionalità non nativa di Kubernetes

Questo diagramma illustra la condivisione del volume NFS tra due namespace Kubernetes.



Avvio rapido

È possibile configurare la condivisione del volume NFS in pochi semplici passaggi.

1 Configurare il PVC di origine per condividere il volume

Il proprietario dello spazio dei nomi di origine concede l'autorizzazione ad accedere ai dati nel PVC di origine.

2 Concedi l'autorizzazione per creare un CR nello spazio dei nomi di destinazione

L'amministratore del cluster concede l'autorizzazione al proprietario dello spazio dei nomi di destinazione per creare il CR TridentVolumeReference.

3 Crea TridentVolumeReference nello spazio dei nomi di destinazione

Il proprietario dello spazio dei nomi di destinazione crea il CR TridentVolumeReference per fare riferimento al PVC di origine.

4 Crea il PVC subordinato nello spazio dei nomi di destinazione

Il proprietario dello spazio dei nomi di destinazione crea il PVC subordinato per utilizzare l'origine dati dal PVC di origine.

Configurare gli spazi dei nomi di origine e di destinazione

Per garantire la sicurezza, la condivisione tra namespace richiede la collaborazione e l'azione del proprietario dello spazio dei nomi di origine, dell'amministratore del cluster e del proprietario dello spazio dei nomi di destinazione. Il ruolo dell'utente viene designato in ogni fase.

Passi

1. **Proprietario dello spazio dei nomi di origine:** Crea il PVC(`pvc1`) nello spazio dei nomi di origine che concede l'autorizzazione alla condivisione con lo spazio dei nomi di destinazione(`namespace2`) utilizzando il `shareToNamespace` annotazione.

```
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 il PV e il suo volume di archiviazione NFS backend.

   - È possibile condividere il PVC con più namespace utilizzando un elenco delimitato da virgole. Per esempio, `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4` .
   - Puoi condividere con tutti gli spazi dei nomi utilizzando `*` . Per esempio, `trident.netapp.io/shareToNamespace: *`
   - È possibile aggiornare il PVC per includere il `shareToNamespace` annotazione in qualsiasi momento.

2. **Amministratore del cluster:** assicurarsi che sia presente il corretto RBAC per concedere l'autorizzazione al proprietario dello spazio dei nomi di destinazione per creare il CR TridentVolumeReference nello spazio dei nomi di destinazione.
3. **Proprietario dello spazio dei nomi di destinazione:** Crea un CR TridentVolumeReference nello spazio dei nomi di destinazione che fa riferimento allo spazio dei nomi di origine `pvc1` .

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Proprietario dello spazio dei nomi di destinazione:** Crea un PVC(`pvc2`) nello spazio dei nomi di destinazione(`namespace2`) utilizzando il `shareFromPVC` annotazione per designare il PVC sorgente.

```
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
```

> La dimensione del PVC di destinazione deve essere inferiore o uguale a quella del PVC di origine.

Risultati

Il Trident legge il `shareFromPVC` annotazione sul PVC di destinazione e crea il PV di destinazione come volume subordinato senza risorse di archiviazione proprie che punta al PV di origine e condivide la risorsa di archiviazione del PV di origine. La destinazione PVC e PV appaiono vincolate normalmente.

Elimina un volume condiviso

È possibile eliminare un volume condiviso tra più namespace. Trident rimuoverà l'accesso al volume nello spazio dei nomi di origine e manterrà l'accesso per gli altri spazi dei nomi che condividono il volume. Quando tutti gli spazi dei nomi che fanno riferimento al volume vengono rimossi, Trident elimina il volume.

Utilizzo `tridentctl get` per interrogare volumi subordinati

Utilizzando il[`tridentctl` utilità, è possibile eseguire il `get` comando per ottenere volumi subordinati. Per maggiori informazioni, fare riferimento al `tridentctl` comandi e opzioni.

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

Bandiere:

- `-h, --help`: Aiuto per i volumi.
- `--parentOfSubordinate string`: Limita la query al volume sorgente subordinato.
- `--subordinateOf string`: Limita la query ai subordinati del volume.

Limitazioni

- Trident non può impedire agli spazi dei nomi di destinazione di scrivere sul volume condiviso. Per evitare la sovrascrittura dei dati del volume condiviso, è opportuno utilizzare il blocco dei file o altri processi.
- Non è possibile revocare l'accesso al PVC di origine rimuovendo il `shareToNamespace` O `shareFromNamespace` annotazioni o eliminazione del `TridentVolumeReference` CR. Per revocare l'accesso, è necessario eliminare il PVC subordinato.
- Non è possibile eseguire snapshot, cloni e mirroring sui volumi subordinati.

Per maggiori informazioni

Per saperne di più sull'accesso ai volumi tra più namespace:

- Visita"Condivisione di volumi tra namespace: dai il benvenuto all'accesso ai volumi tra namespace" .
- Guarda la demo su"NetAppTV" .

Clonazione di volumi tra spazi dei nomi

Utilizzando Trident, puoi creare nuovi volumi utilizzando volumi esistenti o snapshot di volume da uno spazio dei nomi diverso all'interno dello stesso cluster Kubernetes.

Prerequisiti

Prima di clonare i volumi, assicurarsi che i backend di origine e di destinazione siano dello stesso tipo e abbiano la stessa classe di archiviazione.

> La clonazione tra spazi dei nomi è supportata solo per `ontap-san` E `ontap-nas` driver di archiviazione. I cloni di sola lettura non sono supportati.

Avvio rapido

È possibile impostare la clonazione del volume in pochi semplici passaggi.

1. **Configurare il PVC di origine per clonare il volume**

Il proprietario dello spazio dei nomi di origine concede l'autorizzazione ad accedere ai dati nel PVC di origine.

2 Concedi l'autorizzazione per creare un CR nello spazio dei nomi di destinazione

L'amministratore del cluster concede l'autorizzazione al proprietario dello spazio dei nomi di destinazione per creare il CR TridentVolumeReference.

3 Crea TridentVolumeReference nello spazio dei nomi di destinazione

Il proprietario dello spazio dei nomi di destinazione crea il CR TridentVolumeReference per fare riferimento al PVC di origine.

4 Crea il PVC clone nello spazio dei nomi di destinazione

Il proprietario dello spazio dei nomi di destinazione crea PVC per clonare il PVC dallo spazio dei nomi di origine.

Configurare gli spazi dei nomi di origine e di destinazione

Per garantire la sicurezza, la clonazione di volumi tra namespace richiede la collaborazione e l'azione del proprietario del namespace di origine, dell'amministratore del cluster e del proprietario del namespace di destinazione. Il ruolo dell'utente viene designato in ogni fase.

Passi

1. **Proprietario dello spazio dei nomi di origine:** Crea il PVC(`pvc1`) nello spazio dei nomi di origine(`namespace1`) che concede l'autorizzazione alla condivisione con lo spazio dei nomi di destinazione(`namespace2`) utilizzando il `cloneToNamespace` annotazione.

```
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 il PV e il suo volume di archiviazione backend.

- È possibile condividere il PVC con più namespace utilizzando un elenco delimitato da virgole. Per esempio, `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4` .
- Puoi condividere con tutti gli spazi dei nomi utilizzando `*` . Per esempio, `trident.netapp.io/cloneToNamespace: *`
- È possibile aggiornare il PVC per includere il `cloneToNamespace` annotazione in qualsiasi momento.

2. **Amministratore del cluster:** assicurarsi che sia presente il corretto RBAC per concedere l'autorizzazione al proprietario dello spazio dei nomi di destinazione per creare il CR TridentVolumeReference nello spazio dei nomi di destinazione(`namespace2`).
3. **Proprietario dello spazio dei nomi di destinazione:** Crea un CR TridentVolumeReference nello spazio dei nomi di destinazione che fa riferimento allo spazio dei nomi di origine `pvc1` .

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Proprietario dello spazio dei nomi di destinazione:** Crea un PVC(`pvc2`) nello spazio dei nomi di destinazione(`namespace2`) utilizzando il `cloneFromPVC` O `cloneFromSnapshot` , E `cloneFromNamespace` annotazioni per designare il PVC di origine.

```
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
```

Limitazioni

- Per i PVC forniti tramite driver ontap-nas-economy, i cloni di sola lettura non sono supportati.

Replicare i volumi utilizzando SnapMirror

Trident supporta relazioni mirror tra un volume di origine su un cluster e il volume di destinazione sul cluster peer per replicare i dati in caso di ripristino di emergenza. È possibile utilizzare una definizione di risorsa personalizzata (CRD) con namespace, denominata Trident Mirror Relationship (TMR), per eseguire le seguenti operazioni:

- Creare relazioni speculari tra volumi (PVC)
- Rimuovi le relazioni speculari tra i volumi
- Rompere le relazioni speculari
- Promuovere il volume secondario durante le condizioni di disastro (failover)
- Eseguire la transizione senza perdite delle applicazioni da un cluster all'altro (durante i failover o le migrazioni pianificate)

Prerequisiti di replicazione

Prima di iniziare, assicurati che siano soddisfatti i seguenti prerequisiti:

Cluster ONTAP

- * Trident*: Trident versione 22.10 o successiva deve essere presente sia sui cluster Kubernetes di origine che su quelli di destinazione che utilizzano ONTAP come backend.
- **Licenze**: le licenze asincrone ONTAP SnapMirror che utilizzano il bundle Data Protection devono essere abilitate sia sul cluster ONTAP di origine che su quello di destinazione. Fare riferimento a "Panoramica delle licenze SnapMirror in ONTAP" per maggiori informazioni.

  A partire da ONTAP 9.10.1, tutte le licenze vengono fornite come file di licenza NetApp (NLF), ovvero un singolo file che abilita più funzionalità. Fare riferimento a"Licenze incluse con ONTAP One" per maggiori informazioni.

  ⓘ È supportata solo la protezione asincrona SnapMirror .

Sbirciando

- **Cluster e SVM**: i backend di archiviazione ONTAP devono essere peered. Fare riferimento a "Panoramica del peering di cluster e SVM" per maggiori informazioni.

  ⓘ Assicurarsi che i nomi SVM utilizzati nella relazione di replica tra due cluster ONTAP siano univoci.

- * Trident e SVM*: le SVM remote peered devono essere disponibili per Trident sul cluster di destinazione.

Driver supportati

NetApp Trident supporta la replicazione dei volumi con la tecnologia NetApp SnapMirror utilizzando classi di archiviazione supportate dai seguenti driver: **`ontap-nas` : NFS** `ontap-san` : iSCSI **`ontap-san` : FC** `ontap-san` : NVMe/TCP (richiede almeno la versione ONTAP 9.15.1)

La replicazione del volume tramite SnapMirror non è supportata per i sistemi ASA r2. Per informazioni sui sistemi ASA r2, vedere"Scopri di più sui sistemi di archiviazione ASA r2" .

Crea un PVC specchiato

Seguire questi passaggi e utilizzare gli esempi CRD per creare una relazione speculare tra volumi primari e secondari.

Passi

1. Eseguire i seguenti passaggi sul cluster Kubernetes primario:
   a. Creare un oggetto StorageClass con `trident.netapp.io/replication: true` parametro.

      Esempio

      ```
      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. Creare un PVC con StorageClass creato in precedenza.

      Esempio

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

   c. Creare una CR MirrorRelationship con informazioni locali.

Esempio

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

Trident recupera le informazioni interne per il volume e lo stato corrente di protezione dei dati (DP) del volume, quindi popola il campo di stato di MirrorRelationship.

d. Ottenere il CR TridentMirrorRelationship per ottenere il nome interno e l'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. Eseguire i seguenti passaggi sul cluster Kubernetes secondario:

   a. Creare una StorageClass con il parametro trident.netapp.io/replication: true.

Esempio

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

b. Creare una CR MirrorRelationship con informazioni sulla destinazione e sulla sorgente.

Esempio

```
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 creerà una relazione SnapMirror con il nome del criterio di relazione configurato (o predefinito per ONTAP) e la inizializzerà.

c. Creare un PVC con StorageClass creato in precedenza che funga da destinazione secondaria (SnapMirror).

Esempio

```
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 verificherà il CRD TridentMirrorRelationship e non riuscirà a creare il volume se la relazione non esiste. Se la relazione esiste, Trident assicurerà che il nuovo FlexVol volume venga posizionato su una SVM collegata alla SVM remota definita in MirrorRelationship.

Stati di replica del volume

Una relazione Trident Mirror (TMR) è una CRD che rappresenta un'estremità di una relazione di replicazione tra PVC. Il TMR di destinazione ha uno stato che indica a Trident qual è lo stato desiderato. La TMR di destinazione presenta i seguenti stati:

- **Stabilito**: il PVC locale è il volume di destinazione di una relazione speculare e questa è una nuova relazione.
- **Promosso**: il PVC locale è ReadWrite e montabile, senza alcuna relazione mirror attualmente in vigore.
- **Ristabilito**: il PVC locale è il volume di destinazione di una relazione speculare ed era anche precedentemente in quella relazione speculare.
  - Lo stato ristabilito deve essere utilizzato se il volume di destinazione è mai stato in una relazione con il volume di origine, perché sovrascrive il contenuto del volume di destinazione.
  - Lo stato ripristinato non riuscirà se il volume non era precedentemente in relazione con la sorgente.

Promuovere il PVC secondario durante un failover non pianificato

Eseguire il seguente passaggio sul cluster Kubernetes secondario:

- Aggiorna il campo *spec.state* di TridentMirrorRelationship a `promoted` .

Promuovere PVC secondario durante un failover pianificato

Durante un failover pianificato (migrazione), eseguire i seguenti passaggi per promuovere il PVC secondario:

Passi

1. Sul cluster Kubernetes primario, creare uno snapshot del PVC e attendere che venga creato.
2. Sul cluster Kubernetes primario, creare SnapshotInfo CR per ottenere dettagli interni.

   Esempio

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

3. Nel cluster Kubernetes secondario, aggiornare il campo *spec.state* del CR *TridentMirrorRelationship* in *promoted* e *spec.promotedSnapshotHandle* in modo che sia internalName dello snapshot.
4. Nel cluster Kubernetes secondario, confermare lo stato (campo status.state) di TridentMirrorRelationship su promosso.

Ripristinare una relazione mirror dopo un failover

Prima di ripristinare una relazione speculare, scegli il lato che vuoi rendere primario.

Passi

1. Nel cluster Kubernetes secondario, assicurarsi che i valori per il campo *spec.remoteVolumeHandle* su TridentMirrorRelationship siano aggiornati.
2. Sul cluster Kubernetes secondario, aggiorna il campo *spec.mirror* di TridentMirrorRelationship a `reestablished`.

Operazioni aggiuntive

Trident supporta le seguenti operazioni sui volumi primario e secondario:

Replicare il PVC primario in un nuovo PVC secondario

Assicurati di avere già un PVC primario e un PVC secondario.

Passi

1. Eliminare i CRD PersistentVolumeClaim e TridentMirrorRelationship dal cluster secondario (di destinazione) stabilito.
2. Eliminare il CRD TridentMirrorRelationship dal cluster primario (sorgente).
3. Creare un nuovo CRD TridentMirrorRelationship sul cluster primario (sorgente) per il nuovo PVC secondario (destinazione) che si desidera stabilire.

Ridimensiona un PVC specchiato, primario o secondario

Il PVC può essere ridimensionato normalmente, ONTAP espanderà automaticamente tutti i flevxol di destinazione se la quantità di dati supera la dimensione corrente.

Rimuovere la replicazione da un PVC

Per rimuovere la replica, eseguire una delle seguenti operazioni sul volume secondario corrente:

- Eliminare la MirrorRelationship sul PVC secondario. Ciò interrompe la relazione di replicazione.
- Oppure, aggiorna il campo spec.state in *promoted*.

Elimina un PVC (che era stato precedentemente specchiato)

Trident verifica la presenza di PVC replicati e rilascia la relazione di replicazione prima di tentare di eliminare il volume.

Elimina un TMR

L'eliminazione di un TMR su un lato di una relazione speculare fa sì che il TMR rimanente passi allo stato *promosso* prima che Trident completi l'eliminazione. Se il TMR selezionato per l'eliminazione è già nello stato *promosso*, non esiste alcuna relazione mirror e il TMR verrà rimosso e Trident promuoverà il PVC locale a *ReadWrite*. Questa eliminazione rilascia i metadati SnapMirror per il volume locale in ONTAP. Se in futuro questo volume verrà utilizzato in una relazione mirror, dovrà utilizzare un nuovo TMR con uno stato di replica del volume *stabilito* durante la creazione della nuova relazione mirror.

Aggiorna le relazioni mirror quando ONTAP è online

Le relazioni speculari possono essere aggiornate in qualsiasi momento dopo essere state stabilite. Puoi usare il `state: promoted` O `state: reestablished` campi per aggiornare le relazioni. Quando si promuove un volume di destinazione a un normale volume ReadWrite, è possibile utilizzare *promotedSnapshotHandle* per specificare uno snapshot specifico in cui ripristinare il volume corrente.

Aggiorna le relazioni mirror quando ONTAP è offline

È possibile utilizzare un CRD per eseguire un aggiornamento SnapMirror senza che Trident abbia connettività diretta al cluster ONTAP . Fare riferimento al seguente formato di esempio di TridentActionMirrorUpdate:

Esempio

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

`status.state`riflette lo stato del CRD TridentActionMirrorUpdate. Può assumere un valore tra *Riuscito*, *In corso* o *Non riuscito*.

Utilizzare la topologia CSI

Trident può creare e collegare selettivamente volumi ai nodi presenti in un cluster Kubernetes utilizzando "Funzionalità di topologia CSI" .

Panoramica

Utilizzando la funzionalità CSI Topology, l'accesso ai volumi può essere limitato a un sottoinsieme di nodi, in base alle regioni e alle zone di disponibilità. Oggi i provider cloud consentono agli amministratori di Kubernetes di generare nodi basati su zone. I nodi possono essere ubicati in diverse zone di disponibilità all'interno di una regione o tra regioni diverse. Per facilitare il provisioning dei volumi per i carichi di lavoro in un'architettura multizona, Trident utilizza la topologia CSI.

Scopri di più sulla funzionalità CSI Topology "Qui" .

Kubernetes offre due modalità di associazione dei volumi uniche:

- Con `VolumeBindingMode` impostato su `Immediate` Trident crea il volume senza alcuna consapevolezza della topologia. Il binding del volume e il provisioning dinamico vengono gestiti durante la creazione del PVC. Questo è l'impostazione predefinita `VolumeBindingMode` ed è adatto per cluster che non impongono vincoli di topologia. I volumi persistenti vengono creati senza alcuna dipendenza dai requisiti di pianificazione del pod richiedente.
- Con `VolumeBindingMode` impostato su `WaitForFirstConsumer` , la creazione e l'associazione di un volume persistente per un PVC vengono ritardate finché non viene pianificato e creato un pod che utilizza il PVC. In questo modo, i volumi vengono creati per soddisfare i vincoli di pianificazione imposti dai requisiti

di topologia.

> IL `WaitForFirstConsumer` la modalità di associazione non richiede etichette topologiche. Può essere utilizzato indipendentemente dalla funzionalità CSI Topology.

Cosa ti servirà

Per utilizzare la topologia CSI, è necessario quanto segue:

- Un cluster Kubernetes che esegue un"versione Kubernetes supportata"

```
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"}
```

- I nodi nel cluster dovrebbero avere etichette che introducono la consapevolezza della topologia(`topology.kubernetes.io/region` E `topology.kubernetes.io/zone`). Queste etichette **dovrebbero essere presenti sui nodi del cluster** prima dell'installazione di Trident affinché Trident sia a conoscenza della topologia.

```
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"}]
```

Passaggio 1: creare un backend consapevole della topologia

I backend di archiviazione Trident possono essere progettati per effettuare il provisioning selettivo dei volumi in base alle zone di disponibilità. Ogni backend può trasportare un optional `supportedTopologies` blocco che rappresenta un elenco di zone e regioni supportate. Per le StorageClass che utilizzano tale backend, un volume verrà creato solo se richiesto da un'applicazione pianificata in una regione/zona supportata.

Ecco un esempio di definizione del 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`viene utilizzato per fornire un elenco di regioni e zone per backend. Queste regioni e zone rappresentano l'elenco dei valori consentiti che possono essere forniti in una StorageClass. Per le StorageClass che contengono un sottoinsieme delle regioni e delle zone fornite in un backend, Trident crea un volume sul backend.

Puoi definire `supportedTopologies` anche per pool di archiviazione. Vedere il seguente esempio:

```
---
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
```

In questo esempio, il `region` E `zone` le etichette indicano la posizione del pool di archiviazione. `topology.kubernetes.io/region` E `topology.kubernetes.io/zone` determinano da dove possono essere consumati i pool di stoccaggio.

Passaggio 2: definire StorageClass che riconoscono la topologia

In base alle etichette topologiche fornite ai nodi del cluster, è possibile definire StorageClass in modo che contengano informazioni sulla topologia. Ciò determinerà i pool di archiviazione che fungono da candidati per le richieste PVC effettuate e il sottoinsieme di nodi che possono utilizzare i volumi forniti da Trident.

Vedere il seguente esempio:

```
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
```

Nella definizione StorageClass fornita sopra, `volumeBindingMode` è impostato su `WaitForFirstConsumer` . I PVC richiesti con questa StorageClass non verranno elaborati finché non verranno referenziati in un pod. E, `allowedTopologies` fornisce le zone e la regione da utilizzare. IL `netapp-san-us-east1` StorageClass crea PVC su `san-backend-us-east1` backend definito sopra.

Fase 3: creare e utilizzare un PVC

Dopo aver creato StorageClass e mappato un backend, è ora possibile creare PVC.

Vedi l'esempio `spec` sotto:

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

La creazione di un PVC utilizzando questo manifesto produrrebbe quanto segue:

```
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
```

Per far sì Trident crei un volume e lo leghi al PVC, utilizzare il PVC in un contenitore. Vedere il seguente esempio:

```
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
```

Questo podSpec istruisce Kubernetes a pianificare il pod sui nodi presenti nel `us-east1` regione e scegli tra qualsiasi nodo presente nella `us-east1-a` O `us-east1-b` zone.

Vedere il seguente output:

```
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
```

Aggiorna i backend per includere `supportedTopologies`

I backend preesistenti possono essere aggiornati per includere un elenco di `supportedTopologies` usando `tridentctl backend update` . Ciò non influirà sui volumi già forniti e verrà utilizzato solo per i PVC successivi.

Trova maggiori informazioni

- "Gestire le risorse per i contenitori"
- "Selettore di nodi"
- "Affinità e anti-affinità"
- "Contaminazioni e tolleranze"

Lavorare con gli snapshot

Gli snapshot dei volumi persistenti (PV) di Kubernetes consentono copie point-in-time dei volumi. È possibile creare uno snapshot di un volume creato utilizzando Trident, importare uno snapshot creato al di fuori di Trident, creare un nuovo volume da uno snapshot esistente e recuperare i dati del volume dagli snapshot.

Panoramica

Lo snapshot del volume è supportato da `ontap-nas` , `ontap-nas-flexgroup` , `ontap-san` , `ontap-san-economy` , `solidfire-san` , `gcp-cvs` , `azure-netapp-files` , E `google-cloud-netapp-volumes` conducenti.

Prima di iniziare

Per lavorare con gli snapshot è necessario disporre di un controller snapshot esterno e di definizioni di risorse personalizzate (CRD). Questa è responsabilità dell'orchestratore di Kubernetes (ad esempio: Kubeadm, GKE, OpenShift).

Se la distribuzione Kubernetes non include il controller snapshot e i CRD, fare riferimento aDistribuisci un controller di snapshot del volume .

Non creare un controller snapshot se si creano snapshot di volumi on-demand in un ambiente GKE. GKE utilizza un controller snapshot nascosto e integrato.

Crea uno snapshot del volume

Passi

1. Crea un `VolumeSnapshotClass` Per ulteriori informazioni, fare riferimento a"Classe VolumeSnapshot" .
   - IL `driver` indica il pilota Trident CSI.
   - `deletionPolicy`può essere `Delete` O `Retain` . Quando impostato su `Retain` , lo snapshot fisico sottostante sul cluster di archiviazione viene mantenuto anche quando `VolumeSnapshot` l'oggetto viene eliminato.

     Esempio

     ```
     cat snap-sc.yaml
     ```

     ```
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshotClass
     metadata:
       name: csi-snapclass
     driver: csi.trident.netapp.io
     deletionPolicy: Delete
     ```

2. Crea un'istantanea di un PVC esistente.

   Esempi

   - Questo esempio crea un'istantanea di un PVC esistente.

     ```
     cat snap.yaml
     ```

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

   - Questo esempio crea un oggetto snapshot del volume per un PVC denominato `pvc1` e il nome dello snapshot è impostato su `pvc1-snap` . Un VolumeSnapshot è analogo a un PVC ed è associato a un `VolumeSnapshotContent` oggetto che rappresenta l'istantanea effettiva.

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

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

- Puoi identificare il `VolumeSnapshotContent` oggetto per il `pvc1-snap` VolumeSnapshot descrivendolo. IL `Snapshot Content Name` identifica l'oggetto VolumeSnapshotContent che gestisce questo snapshot. IL `Ready To Use` Il parametro indica che lo snapshot può essere utilizzato per creare un nuovo 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
...
```

Creare un PVC da uno snapshot del volume

Puoi usare `dataSource` per creare un PVC utilizzando un VolumeSnapshot denominato `<pvc-name>` come fonte dei dati. Una volta creato, il PVC può essere attaccato a un contenitore e utilizzato come qualsiasi altro PVC.

> Il PVC verrà creato nello stesso backend del volume sorgente. Fare riferimento a"KB: La creazione di un PVC da uno snapshot PVC Trident non può essere creata in un backend alternativo" .

L'esempio seguente crea il PVC utilizzando `pvc1-snap` come fonte dei dati.

```
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
```

Importa uno snapshot del volume

Trident supporta il"Processo di snapshot pre-provisioning di Kubernetes" per consentire all'amministratore del cluster di creare un `VolumeSnapshotContent` oggetti e importare snapshot creati al di fuori di Trident.

Prima di iniziare

Trident deve aver creato o importato il volume padre dello snapshot.

Passi

1. **Amministratore del cluster:** Crea un `VolumeSnapshotContent` oggetto che fa riferimento allo snapshot del backend. In questo modo si avvia il flusso di lavoro degli snapshot in Trident.
   - Specificare il nome dello snapshot del backend in `annotations` COME `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">` .
   - Specificare `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` In `snapshotHandle` Questa è l'unica informazione fornita a Trident dallo snapshotter esterno nel `ListSnapshots` chiamata.

     > IL `<volumeSnapshotContentName>` non può sempre corrispondere al nome dello snapshot backend a causa dei vincoli di denominazione CR.

     Esempio

     L'esempio seguente crea un `VolumeSnapshotContent` oggetto che fa riferimento allo snapshot 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. **Amministratore del cluster:** Crea il `VolumeSnapshot` CR che fa riferimento al `VolumeSnapshotContent` oggetto. Questa richiesta richiede l'accesso per utilizzare il `VolumeSnapshot` in un dato spazio dei nomi.

   Esempio

   L'esempio seguente crea un `VolumeSnapshot` CR nominato `import-snap` che fa riferimento al `VolumeSnapshotContent` nominato `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. **Elaborazione interna (nessuna azione richiesta):** Lo snapshotter esterno riconosce il nuovo creato `VolumeSnapshotContent` e gestisce il `ListSnapshots` chiamata. Trident crea il `TridentSnapshot` .
   - Lo snapshotter esterno imposta il `VolumeSnapshotContent` A `readyToUse` e il `VolumeSnapshot` A `true` .
   - Il ritorno Trident `readyToUse=true` .
4. **Qualsiasi utente:** Crea un `PersistentVolumeClaim` per fare riferimento al nuovo `VolumeSnapshot` , dove il `spec.dataSource` (O `spec.dataSourceRef`) il nome è il `VolumeSnapshot` nome.

Esempio

L'esempio seguente crea un PVC che fa riferimento a `VolumeSnapshot` nominato `import-snap` .

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Recupera i dati del volume utilizzando gli snapshot

La directory snapshot è nascosta per impostazione predefinita per facilitare la massima compatibilità dei volumi forniti tramite `ontap-nas` E `ontap-nas-economy` conducenti. Abilita il `.snapshot` directory per recuperare direttamente i dati dagli snapshot.

Utilizzare il ripristino dello snapshot del volume ONTAP CLI per ripristinare un volume a uno stato registrato in uno snapshot precedente.

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

Quando si ripristina una copia snapshot, la configurazione del volume esistente viene sovrascritta. Le modifiche apportate ai dati del volume dopo la creazione della copia snapshot andranno perse.

Ripristino del volume in loco da uno snapshot

Trident fornisce un rapido ripristino del volume in loco da uno snapshot utilizzando `TridentActionSnapshotRestore` (TASR) CR. Questa CR funziona come un'azione Kubernetes imperativa e non persiste dopo il completamento dell'operazione.

Trident supporta il ripristino degli snapshot su `ontap-san` , `ontap-san-economy` , `ontap-nas` , `ontap-nas-flexgroup` , `azure-netapp-files` , `gcp-cvs` , `google-cloud-netapp-volumes` , E `solidfire-san` conducenti.

Prima di iniziare

È necessario disporre di un PVC associato e di uno snapshot del volume disponibile.

- Verificare che lo stato del PVC sia vincolato.

```
kubectl get pvc
```

- Verificare che lo snapshot del volume sia pronto per l'uso.

```
kubectl get vs
```

Passi

1. Creare il CR TASR. Questo esempio crea un CR per PVC `pvc1` e snapshot del volume `pvc1-snapshot` .

   Il CR TASR deve trovarsi in uno spazio dei nomi in cui sono presenti PVC e 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. Applicare il CR per ripristinare dallo snapshot. Questo esempio ripristina da uno snapshot `pvc1` .

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

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

Risultati

Trident ripristina i dati dallo snapshot. È possibile verificare lo stato di ripristino dello snapshot:

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

- Nella maggior parte dei casi, Trident non riproverà automaticamente l'operazione in caso di errore. Sarà necessario eseguire nuovamente l'operazione.
- Gli utenti Kubernetes senza accesso amministrativo potrebbero dover ottenere l'autorizzazione dall'amministratore per creare un CR TASR nel namespace della propria applicazione.

Elimina un PV con gli snapshot associati

Quando si elimina un volume persistente con snapshot associati, il volume Trident corrispondente viene aggiornato allo "stato di eliminazione". Rimuovere gli snapshot del volume per eliminare il volume Trident .

Distribuisci un controller di snapshot del volume

Se la distribuzione Kubernetes non include il controller snapshot e i CRD, è possibile distribuirli come segue.

Passi

1. Creare CRD di snapshot del volume.

```
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. Creare il controller snapshot.

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

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

Se necessario, aprire `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e aggiornare `namespace` al tuo namespace.

Link correlati

- "Istantanee del volume"
- "Classe VolumeSnapshot"

Lavorare con gli snapshot del gruppo di volumi

Snapshot del gruppo di volumi Kubernetes di volumi persistenti (PV) NetApp Trident offre la possibilità di creare snapshot di più volumi (un gruppo di snapshot di volumi). Questo snapshot del gruppo di volumi rappresenta copie di più volumi eseguite nello stesso momento.

VolumeGroupSnapshot è una funzionalità beta di Kubernetes con API beta. Kubernetes 1.32 è la versione minima richiesta per VolumeGroupSnapshot.

Creare snapshot del gruppo di volumi

L'istantanea del gruppo di volumi è supportata con `ontap-san` driver, solo per protocollo iSCSI, non ancora supportato con Fibre Channel (FCP) né NVMe/TCP. Prima di iniziare

- Assicurati che la versione di Kubernetes sia K8s 1.32 o successiva.
- Per lavorare con gli snapshot è necessario disporre di un controller snapshot esterno e di definizioni di risorse personalizzate (CRD). Questa è responsabilità dell'orchestratore di Kubernetes (ad esempio: Kubeadm, GKE, OpenShift).

  Se la distribuzione Kubernetes non include il controller snapshot esterno e i CRD, fare riferimento aDistribuisci un controller di snapshot del volume .

  > Non creare un controller snapshot se si creano snapshot di gruppi di volumi su richiesta in un ambiente GKE. GKE utilizza un controller snapshot nascosto e integrato.

- Nel controller snapshot YAML, impostare `CSIVolumeGroupSnapshot` feature gate su 'true' per garantire che lo snapshot del gruppo di volumi sia abilitato.
- Creare le classi di snapshot del gruppo di volumi richieste prima di creare uno snapshot del gruppo di volumi.
- Assicurarsi che tutti i PVC/volumi siano sullo stesso SVM per poter creare VolumeGroupSnapshot.

Passi

- Creare una VolumeGroupSnapshotClass prima di creare una VolumeGroupSnapshot. Per maggiori informazioni, fare riferimento a"ClasseSnapshotVolumeGroup" .

```
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
```

- Creare PVC con le etichette richieste utilizzando le classi di archiviazione esistenti oppure aggiungere queste etichette ai PVC esistenti.

  L'esempio seguente crea il PVC utilizzando `pvc1-group-snap` come fonte di dati ed etichetta `consistentGroupSnapshot: groupA` . Definisci la chiave e il valore dell'etichetta in base alle tue esigenze.

```
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 stessa etichetta(`consistentGroupSnapshot: groupA`) specificato nel PVC.

  Questo esempio crea uno snapshot del gruppo di volumi:

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

Recupera i dati del volume utilizzando uno snapshot di gruppo

È possibile ripristinare singoli volumi persistenti utilizzando gli snapshot individuali creati come parte dello snapshot del gruppo di volumi. Non è possibile ripristinare lo snapshot del gruppo di volumi come unità.

Utilizzare il ripristino dello snapshot del volume ONTAP CLI per ripristinare un volume a uno stato registrato in uno snapshot precedente.

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

Quando si ripristina una copia snapshot, la configurazione del volume esistente viene sovrascritta. Le modifiche apportate ai dati del volume dopo la creazione della copia snapshot andranno perse.

Ripristino del volume in loco da uno snapshot

Trident fornisce un rapido ripristino del volume in loco da uno snapshot utilizzando `TridentActionSnapshotRestore` (TASR) CR. Questa CR funziona come un'azione Kubernetes imperativa e non persiste dopo il completamento dell'operazione.

Per maggiori informazioni, vedere "Ripristino del volume in loco da uno snapshot".

Elimina un PV con snapshot di gruppo associati

Quando si elimina uno snapshot del volume di gruppo:

- È possibile eliminare i VolumeGroupSnapshot nel loro complesso, non i singoli snapshot del gruppo.
- Se i PersistentVolume vengono eliminati mentre esiste uno snapshot per quel PersistentVolume, Trident sposterà quel volume in uno stato di "eliminazione" perché lo snapshot deve essere rimosso prima che il volume possa essere rimosso in modo sicuro.
- Se è stato creato un clone utilizzando uno snapshot raggruppato e in seguito il gruppo deve essere eliminato, verrà avviata un'operazione di suddivisione su clone e il gruppo non potrà essere eliminato finché la suddivisione non sarà completata.

Distribuisci un controller di snapshot del volume

Se la distribuzione Kubernetes non include il controller snapshot e i CRD, è possibile distribuirli come segue.

Passi

1. Creare CRD di snapshot del volume.

```
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. Creare il controller snapshot.

```
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
```

Se necessario, aprire `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e aggiornare `namespace` al tuo namespace.

Link correlati

- "ClasseSnapshotVolumeGroup"
- "Istantanee del volume"

Informazioni sul copyright

Informazioni sul marchio commerciale