



Effettua il provisioning e gestisci i volumi

Trident

NetApp
October 29, 2026

Sommario

Effettua il provisioning e gestisci i volumi	1
Esegui il provisioning di un volume Trident	1
Panoramica	1
Crea il PVC	1
Sposta un volume	3
Requisiti	4
Come funziona	4
Sposta un volume	4
Espandi i volumi Trident	7
Espandere un volume iSCSI	7
Espandere un volume FC	10
Espandere un volume NFS	14
Comprendere i limiti del sottosistema RWX NVMe	17
Comprendere il limite di 64 nodi	17
Comprendere i modelli di sottosistema NVMe	17
Identificare i sintomi di errore	18
Risolvere gli errori di limite del sottosistema	18
Aggiorna Trident per applicare il modello super-subsystem	18
Scalabilità del controller	18
Concetti chiave e definizioni	18
Comportamento predefinito (modalità seriale)	19
Supporto per la scalabilità del controller	19
Come funziona enableConcurrency	20
Prima di abilitare	20
Abilita la scalabilità del controller	20
Comportamento della concorrenza	22
Avvertenze e limitazioni	22
Raccomandazioni	23
Espansione automatica del volume	23
Requisiti	23
Limitazioni	24
Provision Volumes con criterio di Autogrow	24
Crea una policy di Autogrow	24
Crea una policy di Autogrow	25
Stati delle policy	26
Associa una policy a un StorageClass	26
Precedenza delle policy	27
Esempi di configurazione	27
Gestisci le policy di Autogrow	30
Visualizza le policy di crescita automatica	30
Aggiorna una policy di Autogrow	30
Elimina un criterio Autogrow	31
Monitorare l'utilizzo della policy Autogrow	32

Protocolli supportati	32
Backend supportati	33
Limitazioni note	33
Domande frequenti	33
Importa i volumi Trident	39
Panoramica e considerazioni	39
Importare un volume	40
Esempi	42
Esempi di ONTAP SAN-economy	47
Personalizza i nomi e le etichette dei volumi di Trident	50
Prima di iniziare	50
Limitazioni	50
Comportamenti chiave dei nomi di volume personalizzabili	51
Esempi di configurazione del backend con name template ed etichette	51
Esempi di template di nome	52
Punti da considerare	52
Condividi un volume NFS Trident tra namespace	53
Caratteristiche	53
Avvio rapido	54
Configurare gli spazi dei nomi di origine e di destinazione	54
Elimina un volume condiviso	56
Utilizzare <code>tridentctl get</code> per interrogare i volumi subordinati	56
Limitazioni	56
Per ulteriori informazioni	56
Clona i volumi Trident tra namespace	56
Prerequisiti	57
Avvio rapido	57
Configurare gli spazi dei nomi di origine e di destinazione	57
Limitazioni	59
Replicare i volumi utilizzando SnapMirror	59
Prerequisiti della replica	59
Crea un PVC specchiato	60
Stati di replicazione del volume	63
Promuovere il PVC secondario durante un failover non pianificato	63
Promuovere il PVC secondario durante un failover pianificato	63
Ripristina una relazione mirror dopo un failover	64
Operazioni aggiuntive	64
Aggiorna le relazioni mirror quando ONTAP è online	65
Aggiorna le relazioni mirror quando ONTAP è offline	65
Usa la topologia CSI	65
Panoramica	65
Passo 1: Creare un backend consapevole della topologia	67
Passo 2: Definire StorageClasses che sono consapevoli della topologia	69
Fase 3: Creare e utilizzare un PVC	69
Aggiorna i backend per includere <code>supportedTopologies</code>	72

Trova ulteriori informazioni	72
Lavora con gli snapshot di Trident	72
Panoramica	72
Crea uno Snapshot del volume	73
Crea un PVC da una Snapshot del volume	74
Importa una Snapshot del volume	75
Recupera i dati del volume utilizzando le Snapshot	77
Ripristino del volume in loco da una Snapshot	77
Eliminare un PV con le relative Snapshot	79
Distribuire un controller snapshot del volume	79
Link correlati	80
Lavora con gli snapshot dei gruppi di volumi di Trident	80
Crea snapshot del gruppo di volumi	80
Recupera i dati del volume utilizzando una Snapshot di gruppo	82
Ripristino del volume in loco da una Snapshot	82
Eliminare un PV con le Snapshot di gruppo associate	82
Distribuire un controller snapshot del volume	83
Link correlati	83

Effettua il provisioning e gestisci i volumi

Esegui il provisioning di un volume Trident

Crea un PersistentVolumeClaim (PVC) che utilizza il StorageClass Kubernetes configurato per richiedere l'accesso al PV. Puoi quindi montare il PV su un pod.

Panoramica

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

Il PVC può essere configurato per richiedere storage di una certa dimensione o modalità di accesso. Utilizzando il StorageClass associato, l'amministratore del cluster può controllare più della sola dimensione e modalità di accesso della PersistentVolume, come ad esempio le prestazioni o il livello di servizio.

Dopo aver creato il PVC puoi montare il volume in un pod.

Crea il PVC

Passaggi

1. Crea 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. Monta il volume in un pod.

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



Puoi monitorare l'avanzamento usando `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 manifest

▼ PersistentVolumeClaim manifesti di esempio

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 un StorageClass denominato 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 un StorageClass denominato protection-gold.

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

▼ Esempi di manifest di pod

Questi esempi mostrano configurazioni di base per collegare 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 di base 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
```

Fate riferimento a ["Oggetti Kubernetes e Trident"](#) per i dettagli su come le classi di storage interagiscono con PersistentVolumeClaim e sui parametri per controllare come Trident effettua il provisioning dei volumi.

Sposta un volume

A partire da Trident 26.06, un amministratore dello storage può spostare in modo non

distruttivo un volume iSCSI su un nodo ONTAP e un aggregato diversi all'interno dello stesso cluster ONTAP mentre il carico di lavoro che lo utilizza continua a funzionare. Richiedi lo spostamento in modo dichiarativo con una `TridentVolumeMove` custom resource o l' `tridentctl` equivalente, e Trident coordina ONTAP e l'host così che l'I/O non venga mai interrotto.

Requisiti

La funzione di spostamento del volume si applica al driver `ontap-san` per i volumi iSCSI solo. I protocolli NVMe e NAS non sono supportati.

Prima di spostare un volume, assicurati che siano soddisfatti i seguenti requisiti:

- Il backend utilizza il `ontap-san` driver con l'API REST ONTAP.
- L'ONTAP cluster ha almeno due nodi.
- Hai uno dei seguenti set di credenziali:
 - ONTAP 9.19.1 o versioni successive se usi il ruolo ONTAP personalizzato dedicato.
 - Credenziali di amministratore del cluster nelle versioni precedenti di ONTAP.

Come funziona

Trident sposta il volume (LUN) tra due posizioni all'interno dello stesso ONTAP cluster. Prima che i dati vengano spostati, Trident aggiunge nuovi percorsi iSCSI alla destinazione, affiancandoli a quelli esistenti, così il dispositivo multipath dell'host non perde mai l'accesso. Dopo che ONTAP ha completato lo spostamento, Trident elimina i percorsi di origine obsoleti.

Trident gestisce lo spostamento attraverso una serie ordinata di stati visibili nella `TridentVolumeMove` risorsa, così puoi seguire l'avanzamento dello spostamento.

Sposta un volume

Identifica il volume e le posizioni

1. Individua il nome del PV da spostare. Il nome del PV è anche il nome della risorsa di spostamento.
2. Dal cluster ONTAP, annota il nodo e l'aggregato correnti (di origine) per quel volume e scegli il nodo e l'aggregato di destinazione.

Inizia lo spostamento

Puoi usare entrambi i metodi. Entrambi creano la stessa `TridentVolumeMove` risorsa, il cui nome è il nome del PV.

Utilizzo di `tridentctl`

```
tridentctl create volume-move \  
  --volume      pvc-8a814d62-bd58-4253-b0d1-82f2885db671 \  
  --source-node ontap-node-01 \  
  --source-pool aggr1_node01 \  
  --target-node ontap-node-02 \  
  --target-pool aggr2_node02
```

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

tvm è un alias per volume-move. Aggiungi `--delete-after-success 10m` per rimuovere automaticamente la risorsa dopo che lo spostamento ha avuto successo.

Utilizzando una risorsa TridentVolumeMove

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

Applica la risorsa:

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

Campi di specifica

sourceNode/ targetNode

Nomi dei nodi ONTAP (non UUID).

sourcePool/ targetPool

Nomi degli aggregate ONTAP.

deleteAfterSuccess

Durata facoltativa, ad esempio 30s o 10m. Imposta su 0 per eliminare subito la risorsa, oppure ometti per conservarla.

Monitora lo spostamento

Monitora i progressi utilizzando il `tvm` nome breve o `tridentvolumemove`:

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

Per visualizzare lo stato attuale e i progressi per ciascun nodo, descrivi la risorsa:

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

Il volume può avere collegamenti simultanei durante lo `Moving` stato, perché ONTAP può impiegare del tempo per completare lo spostamento di un volume di grandi dimensioni. Per garantire uno spostamento senza interruzioni, Trident coordina tra il Trident controller e il Trident node. Ogni stato esegue i seguenti passaggi in ordine, terminando con `Succeeded` o `Failed`.

Stato	Che cosa sta succedendo
Pending	Trident accetta e convalida la richiesta di spostamento e raccoglie le informazioni di cui ha bisogno prima che vengano apportate modifiche allo storage.
ControllerStaging	Trident prepara la posizione di destinazione. Mappa le interfacce di archiviazione del nodo di destinazione sulla LUN e aggiorna Kubernetes in modo che un futuro riavvio del nodo si riconnetta alla posizione corretta.
NodeStaging	Su ciascun host in cui è montato il volume, Trident aggiunge i nuovi percorsi di archiviazione dal nodo di destinazione accanto ai percorsi esistenti, in modo che l'I/O del pod continui su entrambi i set di percorsi.
Moving	ONTAP copia e trasferisce i dati del volume all'aggregato di destinazione. Questa è la fase più lunga e ONTAP la gestisce interamente, mentre Trident esegue il polling fino al completamento. Durante questa fase sono consentite nuove pubblicazioni.
ControllerUnstaging	Il trasferimento dei dati è completato. Trident aggiorna Kubernetes per rimuovere i percorsi di origine dallo stato desiderato e poi scollega le interfacce di storage del nodo di origine dalla LUN.
NodeUnstaging	Su ciascun host, Trident rimuove i percorsi sorgente obsoleti. All'host rimangono solo i percorsi di destinazione e il dispositivo multipath continua a gestire le operazioni di I/O senza interruzioni.
Succeeded	Il volume è completamente spostato e online sulla destinazione. Trident conserva la <code>TridentVolumeMove</code> risorsa oppure la elimina automaticamente se imposti <code>deleteAfterSuccess</code> .
Failed	Si è verificato un errore irreversibile che ha interrotto lo spostamento o ne ha impedito l'avvio. Trident ripristina lo stato del volume per consentire la ripresa delle normali operazioni e mantiene la risorsa con un messaggio di errore. Esamina il messaggio, risolvi la causa principale, elimina la risorsa e riprova.

L' `.status.attachments` elenco riporta lo stato di avanzamento di ciascun nodo collegato nel suo `state` campo:

Stato di collegamento	Significato
Pending	Nodo identificato come collegato; il lavoro non è ancora iniziato.
Bridged	Nuovi percorsi di destinazione aggiunti accanto a quelli esistenti.
Migrated	Il traffico ora scorre sui nuovi percorsi.
Cleaned	Percorsi sorgente obsoleti rimossi dal nodo.

Stato di collegamento	Significato
Detached	Il nodo si è staccato durante lo spostamento ed è stato escluso da ulteriori valutazioni mentre Trident verifica se l'aggancio è pronto per procedere al passo successivo.
Failed	Errore del terminale su questo nodo. Guarda il messaggio.

Pulisci la risorsa di spostamento

Se imposti `deleteAfterSuccess`, Trident rimuove automaticamente la risorsa. Altrimenti, puoi eliminarla manualmente. Prima di richiedere un altro spostamento sullo stesso volume, assicurati che non esista alcuna risorsa `TridentVolumeMove` per quel volume né in uno stato `Succeeded` né in uno stato `Failed`.



Trident non elimina mai automaticamente le operazioni non riuscite. Trident conserva una `Failed` risorsa come record di controllo. Dopo che hai risolto la causa principale, elimina manualmente la risorsa prima di riprovare.

Espandi i volumi Trident

Trident offre agli utenti Kubernetes la possibilità di espandere i propri volumi dopo che sono stati 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 dai `ontap-san`, `ontap-san-economy`, `solidfire-san` driver e richiede Kubernetes 1.16 e versioni successive.

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

Modifica la definizione di `StorageClass` per impostare il campo `allowVolumeExpansion` su `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 un StorageClass già esistente, modificarlo per includere il `allowVolumeExpansion` parametro.

Passaggio 2: crea un PVC con la StorageClass che hai creato

Modifica la definizione del PVC e aggiorna il `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 filesystem.
- Quando si tenta di ridimensionare un PV non collegato, Trident espande il volume sul backend di storage. Dopo che il PVC è stato associato a un pod, Trident esegue una nuova scansione del dispositivo e ridimensiona il filesystem. Kubernetes aggiorna la dimensione del PVC dopo che l'operazione di espansione è stata completata con successo.

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: Espandere il PV

Per ridimensionare il PV creato da 1Gi a 2Gi, modificare la definizione del PVC e aggiornare la `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: convalida l'espansione

È possibile verificare che l'espansione abbia funzionato correttamente controllando le dimensioni del PVC, del PV e del Trident volume:

```

kubectl get pvc san-pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound       pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi
RWO           ontap-san    11m
kubectl get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY STATUS    CLAIM                                STORAGECLASS  REASON  AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi      RWO
Delete              Bound     default/san-pvc                    ontap-san    12m
tridentctl get volumes -n trident
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
+-----+-----+-----+-----+
|          BACKEND UUID   | STATE | MANAGED |
+-----+-----+-----+-----+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san |
| block | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

Espandere un volume FC

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



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

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

Modifica la definizione di StorageClass per impostare il campo `allowVolumeExpansion` su `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 un StorageClass già esistente, modificarlo per includere il `allowVolumeExpansion` parametro.

Passaggio 2: crea un PVC con la StorageClass che hai creato

Modifica la definizione del PVC e aggiorna il `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 per il ridimensionamento di un PV FC:

- 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 filesystem.
- Quando si tenta di ridimensionare un PV non collegato, Trident espande il volume sul backend di storage. Dopo che il PVC è stato associato a un pod, Trident esegue una nuova scansione del dispositivo e ridimensiona il filesystem. Kubernetes aggiorna la dimensione del PVC dopo che l'operazione di espansione è stata completata con successo.

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: Espandere il PV

Per ridimensionare il PV creato da 1Gi a 2Gi, modificare la definizione del PVC e aggiornare la `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: convalida l'espansione

È possibile verificare che l'espansione abbia funzionato correttamente controllando le dimensioni del PVC, del PV e del Trident volume:

```
kubectl get pvc san-pvc
```

NAME	STATUS	VOLUME	CAPACITY
san-pvc	Bound	pvc-8a814d62-bd58-4253-b0d1-82f2885db671	2Gi

```

RWO                ontap-san        11m
kubectl get pv
NAME                                     CAPACITY  ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                STORAGECLASS  REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi        RWO
Delete          Bound    default/san-pvc     ontap-san                12m
tridentctl get volumes -n trident
+-----+-----+-----+
+-----+-----+-----+-----+-----+
|                NAME                |  SIZE  | STORAGE CLASS |
+-----+-----+-----+-----+-----+
|                BACKEND UUID         |  STATE  |  MANAGED  |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san    |
| block      | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true        |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+

```

Espandere un volume NFS

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

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

Per ridimensionare un NFS PV, l'amministratore deve prima configurare la storage class per consentire l'espansione del volume impostando il `allowVolumeExpansion` field su `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: crea un PVC con la StorageClass che hai creato

```
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: Espandere il PV

Per ridimensionare il PV da 20 MiB appena creato a 1 GiB, modifica il PVC e imposta `spec.resources.requests.storage` su 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: convalida 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		

Comprendere i limiti del sottosistema RWX NVMe

ReadWriteMany (RWX) volumi che utilizzano il protocollo NVMe hanno un limite di scalabilità di 64 nodi per volume. Di seguito sono illustrate le limitazioni, viene spiegata l'architettura del sottosistema NVMe coinvolto e vengono delineati i passaggi di risoluzione richiesti.

Comprendere il limite di 64 nodi

Se prevedi di usare i volumi ReadWriteMany (RWX) con il protocollo NVMe, un singolo volume RWX NVMe non può essere montato da più di 64 nodi in un cluster Kubernetes.

Non pianificare carichi di lavoro che montano lo stesso RWX NVMe PersistentVolumeClaim su più di 64 nodi.

Questa limitazione si applica solo ai volumi RWX che utilizzano il protocollo NVMe.

Comprendere i modelli di sottosistema NVMe

Modello di sottosistema per volume (versioni di Trident precedenti alla 26.02)

Nelle versioni di Trident precedenti alla 26.02, i volumi RWX NVMe vengono forniti utilizzando un modello di sottosistema per volume. Ogni volume RWX NVMe è mappato al proprio sottosistema NVMe dedicato su ONTAP.

Questo modello è semplice, ma presenta un limite di scalabilità inferiore. Nei cluster Kubernetes di grandi dimensioni, i limiti del controller del sottosistema vengono raggiunti rapidamente perché ogni volume RWX consuma un sottosistema dedicato.

Modello super-sottosistema (introdotto in Trident 26.02)

A partire da Trident 26.02, i volumi RWX NVMe utilizzano un modello di super-sottosistema condiviso. Più volumi RWX NVMe condividono lo stesso sottosistema NVMe.

Ogni super-sottosistema supporta fino a 1024 namespace (volumi). Questo modello migliora significativamente la scalabilità per i carichi di lavoro RWX e riduce la probabilità di raggiungere i limiti del sottosistema ONTAP.

Ogni volume RWX NVMe supporta fino a 64 nodi.

Identificare i sintomi di errore

Se crei o colleghi volumi RWX NVMe su larga scala, potresti osservare errori simili ai seguenti:

```
Maximum number of controllers reached. No more controllers can be created.
```

Questo errore indica che è stato raggiunto il limite del controller del sottosistema ONTAP NVMe.

Risolvere gli errori di limite del sottosistema

Per superare le limitazioni del sottosistema per volume e sfruttare il modello super-sottosistema, esegui l'aggiornamento a Trident 26.02 o versioni successive.

Aggiorna Trident per applicare il modello super-subsystem

Per applicare il modello super-sottosistema ai volumi RWX NVMe:

1. Aggiorna Trident alla versione 26.02 o successiva.
2. Riduci a zero le repliche di tutti i pod che utilizzano volumi RWX NVMe.
3. Verificare che nessun carico di lavoro stia utilizzando attivamente volumi RWX NVMe.
4. Aumenta nuovamente il numero di pod.

Questa sequenza di riavvio garantisce che i volumi RWX NVMe siano collegati utilizzando il modello super-subsystem.

- Questa limitazione si applica solo ai volumi RWX che utilizzano il protocollo NVMe.
- Il limite di 64 nodi si applica per ogni volume RWX NVMe.
- Altre modalità di accesso e altri protocolli non sono interessati.

Scalabilità del controller

Trident migliora la scalabilità del controller grazie a una maggiore concorrenza tra più driver di storage. Quando abiliti la scalabilità del controller, il controller Trident elabora le operazioni di storage in parallelo invece di serializzarle, il che aumenta il throughput negli ambienti Kubernetes con molte operazioni simultanee.

Prima di distribuire, determina quali driver Trident supportano la scalabilità del controller nella disponibilità generale e quali driver sono disponibili come anteprima tecnica in Trident 26.06. Questo ti aiuta a prendere decisioni di distribuzione informate e a gestire il rischio. La scalabilità del controller è disabilitata per impostazione predefinita.

Concetti chiave e definizioni

Scalabilità del controller

La scalabilità del controller si riferisce alla capacità del controller Trident di elaborare più operazioni di storage in parallelo, anziché serializzarle dietro un singolo blocco. Queste operazioni includono la creazione, l'eliminazione e il ridimensionamento dei volumi; la creazione e l'eliminazione di snapshot; la pubblicazione e

l'annullamento della pubblicazione dei volumi; e la gestione del backend.

Quando abiliti la scalabilità del controller, le operazioni su volumi e backend diversi procedono contemporaneamente. Questo aumenta il throughput e riduce il tempo di esecuzione end-to-end in ambienti con un elevato numero di operazioni PersistentVolumeClaim e VolumeSnapshot simultanee.

Comportamento predefinito (modalità seriale)

Per impostazione predefinita, il controller Trident elabora le operazioni una alla volta. Ogni richiesta di creazione, eliminazione, ridimensionamento o snapshot viene completata prima che Trident avvii la successiva. La modalità seriale è quella predefinita supportata per tutte le installazioni e non richiede configurazione.

La modalità seriale è sufficiente per la maggior parte dei carichi di lavoro. Abilita la scalabilità del controller solo quando il volume delle operazioni crea un accumulo durante l'elaborazione in modalità seriale.

Supporto per la scalabilità del controller

Trident supporta la scalabilità del controller a diversi livelli di maturità per diversi driver di storage.

Disponibilità generale

I seguenti driver supportano la scalabilità del controller alla disponibilità generale in Trident 26.06:

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



I `google-cloud-netapp-volumes` e `google-cloud-netapp-volumes-san` driver sono diversi. Trident supporta solo `google-cloud-netapp-volumes`. Non usare `google-cloud-netapp-volumes-san` nelle configurazioni o negli esempi di backend.

Anteprima tecnica

Il seguente driver supporta la scalabilità del controller come funzionalità in anteprima tecnica in Trident 26.06:

- `asa-r2` (SAN e NVMe)

Questo driver presenta le seguenti limitazioni:

- La funzionalità di concorrenza del controller è disponibile solo per valutazione e test.
- Il comportamento può cambiare da una versione all'altra.
- NetApp non consiglia l'utilizzo in ambienti di produzione.

Come funziona enableConcurrency

Quando imposti `enableConcurrency` su `true`, Trident applica l'elaborazione simultanea a tutti i backend gestiti dal controller. L'impostazione si applica a tutti i backend contemporaneamente. Non puoi abilitarla per singoli backend o singoli driver.

Ogni backend configurato deve utilizzare un driver presente nell'elenco di disponibilità generale o di anteprima tecnica. Se un backend utilizza un driver non supportato, Trident non si avvia e Trident non aggiunge un backend che utilizza un driver non supportato.

Per ripristinare l'avvio, rimuovi o riconfigura qualsiasi backend che utilizza un driver non supportato oppure disabilita la scalabilità del controller.

Prima di abilitare



Prima di abilitare la scalabilità del controller, assicurati che ogni backend configurato utilizzi un driver presente nell'elenco di disponibilità generale o di anteprima tecnica. Se un backend utilizza un driver non supportato, Trident non si avvia dopo che hai abilitato la funzionalità.

Utilizza la tabella seguente per decidere se abilitare la scalabilità del controller.

Se il tuo ambiente	Poi
Utilizzi solo driver supportati e riscontri un accumulo di operazioni del controller	Abilita la scalabilità del controller.
Utilizza un driver non supportato	Mantieni la modalità seriale predefinita. Non abilitare la scalabilità del controller.
Gestisce bassi volumi di operazioni senza arretrati	Mantieni la modalità seriale predefinita.

Abilita la scalabilità del controller

L' `enableConcurrency` opzione di configurazione controlla la scalabilità del controller. Devi abilitare esplicitamente questa opzione durante l'installazione di Trident o quando aggiorni una distribuzione esistente.

Distribuzione dell'operatore Trident

Per abilitare la scalabilità del controller con l'operatore Trident, imposta `enableConcurrency` su `true` nella `TridentOrchestrator` risorsa personalizzata (CR).

Nuova installazione

Crea o modifica il `TridentOrchestrator` CR e imposta `enableConcurrency` su `true`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
```

```
enableConcurrency: true
```

Applica il CR:

```
kubectl apply -f tridentorchestrator_cr.yaml
```

Installazione esistente

Applicare una patch al TridentOrchestrator CR esistente per abilitare la scalabilità del controller:

```
kubectl patch torc trident --type=merge -p  
'{"spec":{"enableConcurrency":true}}'
```

Verifica che Trident abbia applicato l'impostazione:

```
kubectl get torc trident -o  
jsonpath='{.status.currentInstallationParams.enableConcurrency}'
```

Distribuzione Helm

Per abilitare la scalabilità del controller con Helm, impostare il enableConcurrency valore su true.

Nuova installazione

```
helm install trident netapp-trident/trident-operator --namespace trident  
--create-namespace --set enableConcurrency=true
```

Installazione esistente

```
helm upgrade trident netapp-trident/trident-operator --namespace trident  
--set enableConcurrency=true
```

In alternativa, imposta enableConcurrency su true in un file values.yaml personalizzato:

```
# values.yaml  
enableConcurrency: true
```

Quindi installa o aggiorna utilizzando il file dei valori:

```
helm install trident netapp-trident/trident-operator --namespace trident
```

```
--create-namespace -f values.yaml
```

implementazione tridentctl

Per abilitare la scalabilità del controller con `tridentctl`, passare il `--enable-concurrency` flag durante l'installazione.

Nuova installazione

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

Installazione esistente

Per abilitare la scalabilità del controller su una distribuzione `tridentctl` esistente, disinstalla Trident e reinstallalo con il flag:

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

Verifica che la scalabilità del controller sia abilitata

Dopo aver abilitato la scalabilità del controller, verifica che il controller Trident sia in esecuzione con la concorrenza abilitata. Controlla i log del pod del controller:

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

L'output include una voce di registro che conferma che la concorrenza è abilitata.

Comportamento della concorrenza

Quando la scalabilità del controller è abilitata, il controller Trident adotta il seguente comportamento:

- Trident sostituisce il blocco globale singolo con un blocco granulare per risorsa.
- Trident serializza le operazioni che modificano la stessa risorsa per mantenere la coerenza dei dati.
- Le operazioni che si limitano a leggere da una risorsa procedono contemporaneamente ad altre operazioni di lettura su quella risorsa.
- Trident limita le richieste API ONTAP simultanee a 20 per LIF di gestione per evitare il sovraccarico dei sistemi di storage di backend.
- Se più backend condividono lo stesso LIF di gestione, condividono questo limite di 20 richieste.

Avvertenze e limitazioni

Le seguenti considerazioni si applicano alla scalabilità del controller in Trident 26.06:

- La scalabilità del controller supporta solo i driver presenti negli elenchi di disponibilità generale e di anteprima tecnica. Per i dettagli, vedi [Prima di abilitare](#).

- Il controller Trident gestisce la concorrenza internamente. Questa versione non prevede limiti di concorrenza configurabili dall'utente.
- Il throughput complessivo dipende dal driver di storage in uso, dalla reattività del backend e dalle prestazioni del server API di Kubernetes.
- Un'elevata concorrenza può aumentare il carico sui sistemi di storage back-end.
- Il comportamento di scalabilità del controller non è identico per tutti i driver.
- Il driver di anteprima tecnica può presentare prestazioni incoerenti sotto carico elevato e può cambiare comportamento tra una versione e l'altra.
- Il debug di operazioni concorrenti può risultare più complesso a causa dell'esecuzione parallela. Metriche e log possono mostrare l'output delle operazioni interlacciate.

Raccomandazioni

Prima di abilitare la scalabilità del controller, completa i seguenti passaggi:

1. Conferma che ogni backend configurato utilizza un driver presente nell'elenco di disponibilità generale o di anteprima tecnica.
2. Testa la modifica in un cluster non di produzione prima di applicarla in produzione.
3. Verifica che la scalabilità del controller sia abilitata dopo aver applicato la modifica.

Applica le seguenti raccomandazioni generali quando lavori con la scalabilità del controller:

- Usa i driver di disponibilità generale per gli ambienti di produzione che richiedono un'elevata scalabilità.
- Valuta il driver di anteprima tecnica solo in ambienti non di produzione.
- Monitora le prestazioni del backend e del controller quando operi su larga scala.
- Non dare per scontato l'ordine delle operazioni negli script di automazione.

Espansione automatica del volume

L'espansione automatica del volume consente ai Persistent Volumes forniti da Trident di aumentare automaticamente quando la capacità utilizzata raggiunge una soglia definita. Questa funzionalità riduce il sovraccarico operativo e aiuta a prevenire l'interruzione delle applicazioni causata dall'esaurimento della capacità. L'espansione automatica del volume viene implementata utilizzando Autogrow Policies. Una Autogrow Policy definisce:

- La soglia di utilizzo che attiva l'espansione
- La quantità di cui cresce il volume
- La dimensione massima che il volume può raggiungere



I volumi aumentano automaticamente di dimensione quando viene superata la soglia di utilizzo definita. I volumi non vengono mai ridotti automaticamente.

Requisiti

Prima di configurare l'espansione automatica del volume, assicurarsi che siano soddisfatti i seguenti requisiti:

- Trident 26.02 o versioni successive
- Autorizzazioni di controllo degli accessi basate sui ruoli per creare `TridentAutogrowPolicy` risorse personalizzate
- `StorageClasses` configurati con `allowVolumeExpansion: true`
- Protocolli ONTAP supportati:
 - Network File System (NFS)
 - Internet Small Computer Systems Interface (iSCSI)
 - Non-Volatile Memory Express (NVMe)
 - Protocollo Fibre Channel (FCP)

Limitazioni

- I volumi raw block ONTAP Non-Volatile Memory Express precedenti a ONTAP 9.16.1 non supportano l'espansione automatica.
- Per i volumi di storage area network, se `growthAmount` è inferiore o uguale a 50 mebibyte, Trident aumenta automaticamente il valore a 51 mebibyte prima del ridimensionamento, a condizione che la dimensione risultante non superi `maxSize`.
- Negli ambienti brownfield, l'espansione automatica potrebbe non funzionare per alcuni volumi esistenti a causa del comportamento di migrazione della pubblicazione del volume.
- Quando un volume raggiunge `maxSize`, non si verifica alcuna ulteriore espansione.
- Protocolli supportati per l'espansione automatica del volume:
 - Network File System (NFS)
 - Internet Small Computer Systems Interface (iSCSI)
 - Non-Volatile Memory Express (NVMe)
 - Protocollo Fibre Channel (FCP)

Provision Volumes con criterio di Autogrow

La policy di Autogrow può essere configurata a due livelli:

- Livello di classe di archiviazione: imposta il valore predefinito per tutti i volumi (usando l'annotazione)
- Livello PVC: sostituisce la classe di archiviazione predefinita (utilizzando annotazione)

Crea una policy di Autogrow

Le Autogrow Policies consentono l'espansione automatica del volume quando i volumi raggiungono una soglia di capacità definita.

Assicurati di avere:

- Trident 26.02 o versioni successive installato
- Autorizzazioni di controllo degli accessi basate sui ruoli per creare `TridentAutogrowPolicy` risorse
- Comprensione dei requisiti di crescita del carico di lavoro

Una Autogrow Policy definisce come i volumi si espandono automaticamente quando raggiungono una soglia

di capacità definita.

È possibile creare Autogrow Policies in qualsiasi momento del flusso di lavoro:

- Prima che StorageClasses e i volumi vengano creati
- Dopo che StorageClasses esistono
- Dopo che i volumi sono stati effettuati il provisioning

Questa flessibilità consente di introdurre l'espansione automatica senza ricreare le risorse esistenti.

Specifiche della Autogrow Policy

Le Autogrow Policies sono risorse personalizzate di Kubernetes definite come segue:

Campo	Descrizione	Formato	Richiesto	Esempio	Predefinito
nome	Identificatore univoco della policy	Stringa	Sì	production-db-policy	Nessuno
usedThreshold	Percentuale di capacità che innesca l'espansione	Stringa percentuale	Sì	"80%"	Nessuno
growthAmount	Quantità da aumentare quando viene raggiunta la soglia	Percentuale o dimensione	No	"10%" o "5Gi"	"10%"
maxSize	Limite massimo della dimensione del volume	Quantità Kubernetes	No	"500Gi"	Illimitato

Crea una policy di Autogrow

Passaggi

1. Crea un file YAML che definisce la tua Autogrow Policy:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

2. Applica la policy al tuo cluster:

```
kubectl apply -f autogrow-policy.yaml
```

3. Verificare che la policy sia stata creata:

```
kubectl get tridentautogrowpolicy standard-autogrow
```

Output previsto

NAME	USED THRESHOLD	GROWTH AMOUNT	STATE
standard-autogrow	80%	10%	Success

Stati delle policy

Dopo aver creato una policy, Trident convalida la specifica e assegna uno dei seguenti stati:

Stato	Descrizione	Azione richiesta
Successo	La policy è convalidata e pronta per l'uso.	Nessuno.
Fallito	Sono stati rilevati errori di convalida.	Rivedi e correggi la specifica.
Eliminazione	L'eliminazione è in corso.	Attendere il completamento.

Associa una policy a un StorageClass

È possibile associare una Autogrow Policy a una StorageClass utilizzando l'annotazione `trident.netapp.io/autogrowPolicy`. Tutti i volumi forniti da quella StorageClass ereditano la policy.



Il StorageClass deve avere `allowVolumeExpansion: true`.

Passaggi

1. Crea o modifica una StorageClass con l'annotazione Autogrow Policy:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

2. Applica il StorageClass:

```
kubectl apply -f storageclass.yaml
```

3. Verifica l'annotazione:

```
kubectl get storageclass ontap-gold -o  
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Output previsto

```
production-db-policy
```

Precedenza delle policy

Quando le annotazioni Autogrow Policy sono impostate sia su una StorageClass che su un PVC, Trident applica le seguenti regole di precedenza:

1. **L'annotazione PVC ha la priorità.** Se un PVC imposta `trident.netapp.io/autogrowPolicy`, quel valore viene sempre utilizzato.
2. **StorageClass annotation si applica solo quando il PVC non ha annotazione.**
3. **Se nessuno dei due ha l'annotazione,** non viene applicata alcuna Autogrow Policy.

Annotazione StorageClass	Annotazione PVC	Comportamento efficace
<code>trident.netapp.io/autogrowPolicy: standard-agp</code>	Non impostato	Usi standard-agp.
<code>trident.netapp.io/autogrowPolicy: standard-agp</code>	<code>trident.netapp.io/autogrowPolicy: logs-policy</code>	Utilizza logs-policy (PVC sostituisce StorageClass).
<code>trident.netapp.io/autogrowPolicy: standard-agp</code>	<code>trident.netapp.io/autogrowPolicy: "none"</code>	No Autogrow Policy (PVC disabilita autogrow).
Non impostato	<code>trident.netapp.io/autogrowPolicy: dev-policy</code>	Usi dev-policy.
Non impostato	Non impostato	Nessuna Autogrow Policy.

Esempi di configurazione

Gli esempi seguenti mostrano configurazioni comuni di Autogrow Policy per diversi casi d'uso.

Politica conservativa per database di produzione

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

Archiviazione dei log con incrementi di crescita fissi

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: log-storage-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Politica minima con valori predefiniti

Quando si omettono `growthAmount` e `maxSize`, Trident utilizza i valori predefiniti (10% crescita, dimensione illimitata):

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: logs-policy
spec:
  usedThreshold: "85%"
```

Policy con un `maxSize` personalizzato e `growthAmount` predefinito

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: default-ga-policy
spec:
  usedThreshold: "70%"
  maxSize: "100Gi"
```

Crescita aggressiva con maxSize illimitata

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

Policy con percentuali frazionarie

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: precise-policy
spec:
  usedThreshold: "80.28%"
  growthAmount: "10.65%"
  maxSize: "100Gi"
```



Sono supportate le percentuali frazionarie. Se si specificano più di tre cifre decimali, Trident arrotonda il valore a tre cifre decimali.

NAS StorageClass con crescita automatica

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN StorageClass con crescita automatica

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
```

```
name: database-storage
annotations:
  trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Gestisci le policy di Autogrow

Dopo che crei le Autogrow Policies, puoi visualizzarle, aggiornarle ed eliminarle. Puoi anche monitorare quali volumi usano una determinata policy.

Visualizza le policy di crescita automatica

Elenca tutte le politiche

Usa `kubectl` per elencare tutte le Autogrow Policies nel tuo cluster:

```
kubectl get tridentautogrowpolicy
```

In alternativa, usa `tridentctl`:

```
tridentctl get autogrowpolicy
```

Visualizza dettagli della polizza

Per visualizzare le specifiche complete e lo stato di una policy:

```
kubectl describe tridentautogrowpolicy production-db-policy
```

Per visualizzare una policy con i volumi associati in formato YAML:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Aggiorna una policy di Autogrow

È possibile modificare una policy esistente per cambiarne la soglia, l'entità della crescita o la dimensione massima. Le modifiche hanno effetto immediato per tutti i volumi che utilizzano la policy.



Le modifiche influiscono su tutti i volumi che utilizzano la policy. Quando possibile, testa prima le

modifiche in un ambiente non di produzione.

Passaggi

1. Modifica la policy:

```
kubectl edit tridentautogrowpolicy production-db-policy
```

2. Modifica i spec campi secondo necessità:

```
spec:
  usedThreshold: "75%"      # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

3. Salva ed esci. Le modifiche hanno effetto immediato.

Considerazioni sull'aggiornamento

- **Effetto immediato:** Tutti i volumi che utilizzano la policy adottano i nuovi parametri alla successiva valutazione della crescita.
- **Non è necessario riavviare il volume:** Le modifiche si applicano alla prossima operazione di crescita.
- **Test first:** convalidare le modifiche in un ambiente non di produzione quando possibile.
- **Comunica le modifiche:** Notifica ai team quando modifichi le policy condivise.

Elimina un criterio Autogrow

Le policy di crescita automatica utilizzano la protezione del finalizzatore per impedire la cancellazione accidentale mentre i volumi le utilizzano attivamente.

Passaggi

1. Elimina la policy:

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

2. Se i volumi utilizzano ancora la policy, l'eliminazione entra in uno stato `Deleting`. Verifica quali volumi sono interessati:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

3. Rimuovi il criterio da ogni volume interessato. Scegli una delle seguenti opzioni:

- **Opzione A: Disattivare esplicitamente l'autogrow** impostando l'annotazione su `"none"`:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

◦ **Opzione B: Rimuovi completamente l'annotazione:**

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Comportamento di eliminazione

Scenario	Comportamento
Nessun volume utilizza la policy	La policy viene eliminata immediatamente.
I volumi utilizzano la policy	La policy entra <code>Deleting</code> nello stato. Un finalizzatore blocca il completamento finché tutti i volumi non vengono rimossi.
Tutti i volumi vengono rimossi dalla policy	I finalizzatori vengono rimossi e la policy viene eliminata.

Monitorare l'utilizzo della policy Autogrow

Verificare i volumi utilizzando una policy

```
tridentctl get autogrowpolicy production-db-policy -o json | jq '.volumes'
```

Scopri quale policy utilizza un volume

```
kubectl get pvc database-pvc -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Monitorare gli eventi delle policy

```
kubectl get events --field-selector
involvedObject.kind=TridentAutogrowPolicy
```

Protocolli supportati

Autogrow supporta i seguenti protocolli di archiviazione:

- NFS
- iSCSI

- FCP
- NVMe



Per i volumi SAN, se la `growthAmount` configurata è pari o inferiore a 50 MiB, Trident aumenta automaticamente la quantità di crescita a 51 MB per l'operazione di ridimensionamento, a condizione che la dimensione risultante non superi `maxSize`.

Backend supportati

Autogrow supporta i backend ONTAP. Autogrow supporta anche Google Cloud NetApp Volumes (GCNV) sia per i volumi NAS che SAN.

Limitazioni note

- **ONTAP NVMe raw block volumes:** I volumi creati con versioni di ONTAP precedenti alla 9.16.1 non supportano l'autogrow.
- **Volumi esistenti (implementazioni brownfield):** L'espansione automatica potrebbe non funzionare per i volumi esistenti anche se è stata applicata una Autogrow Policy valida. Questa limitazione è dovuta a una migrazione in corso delle pubblicazioni dei volumi. Per verificare che la migrazione sia completa, controlla i log del controller Trident per messaggi "Migration completed".

Domande frequenti

Quando Trident valuta la soglia?

Trident monitora costantemente l'utilizzo del volume. Quando la capacità utilizzata supera il `usedThreshold`, Trident crea una richiesta di ridimensionamento interna ed espande il volume della `growthAmount` configurata.

Ad esempio, questa policy attiva l'espansione all'80% della capacità e aumenta il volume del 10% ogni volta, fino a un massimo di 500 GiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

Posso applicare una policy dopo che i volumi sono già stati forniti?

Sì. È possibile creare una Autogrow Policy in qualsiasi momento e applicarla ai PVC esistenti aggiungendo o aggiornando l'annotazione `trident.netapp.io/autogrowPolicy`. Non è necessario ricreare il PVC o il StorageClass.

Per applicare una policy a un PVC esistente:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Per applicare una policy a un'istanza esistente di StorageClass:

```
kubectl annotate storageclass ontap-gold \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Cosa succede se imposto una politica di crescita automatica sia sulla StorageClass che sul PVC?

L'annotazione PVC ha sempre la precedenza. Se un PVC ha l'annotazione `trident.netapp.io/autogrowPolicy`, Trident utilizza quel valore indipendentemente da quanto specificato in StorageClass. Per i dettagli, fai riferimento a ["Precedenza delle policy"](#).

Ad esempio, dato questo StorageClass:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

E questo PVC che annulla la politica StorageClass:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
  storageClassName: ontap-gold
```

Trident usa `logs-policy` per `database-pvc`, non `standard-agp`.

Come faccio a disattivare l'autogrow per un volume specifico?

Imposta l'annotazione PVC su `"none"`. Questa impostazione sovrascrive qualsiasi policy a livello di StorageClass per quel volume:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

Puoi verificare che autogrow sia disabilitato:

```
kubectl get pvc <pvc-name> -o jsonpath
='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Output previsto

```
none
```

Cosa succede quando un volume raggiunge maxSize?

Trident interrompe l'espansione del volume. Non crea ulteriori richieste di ridimensionamento per quel volume, anche se l'utilizzo continua ad aumentare oltre il `usedThreshold`.

Ad esempio, con questa policy, Trident interrompe la crescita del volume dopo che raggiunge i 100 GiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: capped-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Per consentire una crescita illimitata, ometti `maxSize` o impostalo su 0:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: unlimited-policy
spec:
  usedThreshold: "85%"
```

```
growthAmount: "10%"
```

Posso modificare una policy senza riavviare i volumi?

Sì. Quando aggiorni una policy, tutti i volumi che utilizzano quella policy adottano i nuovi parametri alla successiva valutazione della crescita. Non è necessario riavviare i volumi.

Per aggiornare una policy in vigore:

```
kubectrl edit tridentautogrowpolicy production-db-policy
```

Modifica i campi secondo necessità:

```
spec:
  usedThreshold: "75%"      # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

Salva ed esci. Verifica la policy aggiornata:

```
kubectrl get tridentautogrowpolicy production-db-policy
```

Output previsto

NAME	USED THRESHOLD	GROWTH AMOUNT	STATE
production-db-policy	75%	20%	Success

Perché la mia policy è in stato Failed?

Uno `Failed` stato indica che la specifica della policy contiene errori di convalida. Esegui il seguente comando per visualizzare i dettagli dell'errore:

```
kubectrl describe tridentautogrowpolicy <policy-name>
```

Le cause più comuni includono un `usedThreshold` non valido (deve essere compreso tra 1 e 99%), un `growthAmount` che supera `maxSize`, o un formato di quantità Kubernetes non valido. Correggere la specifica e riapplicarla:

```
kubectrl apply -f autogrow-policy.yaml
```

Perché non posso eliminare una policy?

Le policy utilizzano la protezione del finalizzatore. Se i volumi utilizzano ancora la policy, l'eliminazione entra in uno stato `Deleting` e attende finché non rimuovi tutti i volumi dalla policy.

Identificare i volumi interessati:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Quindi rimuovere l'annotazione da ogni PVC:

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Dopo aver rimosso tutti i volumi, Trident rilascia il finalizzatore ed elimina la policy.

Quali backend sono supportati da autogrow?

Autogrow supporta i backend ONTAP. Supporta inoltre Google Cloud NetApp Volumes (GCNV) sia per i volumi NAS che SAN.

Autogrow supporta i protocolli NFS, iSCSI, FCP e NVMe. I volumi a blocchi raw NVMe richiedono ONTAP 9.16.1 o versioni successive.

I volumi esistenti nelle implementazioni brownfield potrebbero richiedere il completamento della migrazione della pubblicazione dei volumi prima che l'espansione automatica abbia effetto. Per verificare lo stato della migrazione, controlla i log del controller Trident:

```
kubectl logs -l app=trident-controller -n trident | grep "Migration completed"
```

I seguenti esempi di StorageClass mostrano l'autogrow configurato per i backend NAS e SAN:

Backend NAS

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
```

```

provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true

```

Backend SAN

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true

```

Qual è la quantità minima di crescita per i volumi SAN?

Per i volumi SAN, la crescita minima effettiva è di 51 MB. Se si configura un `growthAmount` di 50 MiB o meno, Trident aumenta automaticamente la crescita a 51 MB per l'operazione di ridimensionamento.

Ad esempio, questa policy imposta una `growthAmount` di "40Mi", ma Trident applica una crescita di 51 MB per qualsiasi volume SAN che la utilizza:

```

apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-minimal-policy
spec:
  usedThreshold: "85%"
  growthAmount: "40Mi"
  maxSize: "100Gi"

```

Per evitare questa regolazione automatica, impostare `growthAmount` su un valore maggiore di 50 MiB:

```

apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-policy
spec:
  usedThreshold: "85%"

```

```
growthAmount: "100Mi"  
maxSize: "500Gi"
```

Importa i volumi Trident

È possibile importare volumi di archiviazione esistenti come PV Kubernetes utilizzando `tridentctl import` o creando un Persistent Volume Claim (PVC) con annotazioni di importazione Trident.

Panoramica e considerazioni

È possibile importare un volume in Trident per:

- Containerizzare un'applicazione e riutilizzare il set di dati esistente
- Utilizza un clone di un set di dati per un'applicazione effimera
- Ricostruire un cluster Kubernetes guasto
- Migrare i dati delle applicazioni durante il disaster recovery

Considerazioni

Prima di importare un volume, esaminare le seguenti considerazioni.

- Trident può importare solo volumi ONTAP di tipo RW (read-write). I volumi di tipo DP (data protection) sono volumi di destinazione di SnapMirror. È necessario interrompere la relazione di mirroring prima di importare il volume in Trident.
- Si consiglia di importare volumi senza connessioni attive. Per importare un volume utilizzato attivamente, clonare il volume e poi eseguire l'importazione.



Questo 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. Questo può causare la corruzione dei dati.

- Sebbene `StorageClass` debba 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, durante l'importazione non è richiesta la selezione del pool. Pertanto, l'importazione non fallirà anche se il volume esiste su un backend o un 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 esegue correttamente il binding del PVC e del PV, la politica di recupero viene aggiornata per corrispondere alla politica di recupero della Storage Class.
 - Se la reclaim policy della Storage Class è `delete`, Trident elimina il volume di storage quando il PV viene eliminato.
- Per impostazione predefinita, Trident gestisce il PVC e rinomina il FlexVol volume e la LUN o qtree sul backend. Puoi passare il `--no-manage` flag per importare un volume non gestito e il `--no-rename` flag per mantenere il nome del volume.

- `--no-manage*` - Se si utilizza il `--no-manage` flag, Trident non esegue alcuna operazione aggiuntiva sul PVC o sul PV per il ciclo di vita degli oggetti. Il volume di archiviazione non viene eliminato quando il PV viene eliminato e altre operazioni come il clone del volume e il ridimensionamento del volume vengono anch'esse ignorate.
- `--no-rename*` - Se usi il `--no-rename` flag, Trident mantiene il nome del volume esistente durante l'importazione dei volumi e gestisce il ciclo di vita dei volumi. Questa opzione è supportata solo per i `ontap-nas`, `ontap-san` (inclusi i sistemi ASA r2), `ontap-san-economy` e `ontap-nas-economy` (importazione `qtree`) driver.



Queste opzioni sono utili se si desidera utilizzare Kubernetes per i carichi di lavoro containerizzati, ma altrimenti si desidera gestire il ciclo di vita del volume di archiviazione al di fuori di Kubernetes.

- Al PVC e al PV viene aggiunta un'annotazione 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.

Importare un volume

È possibile importare un volume utilizzando `tridentctl import` oppure creando un PVC con annotazioni di importazione Trident.



Se si utilizzano annotazioni PVC, non è necessario scaricare o utilizzare `tridentctl` per importare il volume.

Utilizzo di `tridentctl`

Passaggi

1. Crea un file 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 definizione del 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
```



Includere solo i parametri obbligatori. Parametri aggiuntivi come il nome del PV o la dimensione del volume possono causare il fallimento del comando di importazione.

2. Usa il comando `tridentctl import` per specificare il nome del backend Trident contenente il volume e il nome che identifica in modo univoco il volume sullo storage (ad esempio: ONTAP FlexVol,

Element Volume). L'argomento `-f` è necessario per specificare il percorso del file PVC.

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

Utilizzo delle annotazioni PVC

Passaggi

1. Creare un file PVC YAML (ad esempio, `pvc.yaml`) con le annotazioni di importazione Trident richieste. Il file PVC deve includere:

- `name` and `namespace` nei metadati
- `accessModes`, `resources.requests.storage`, e `storageClassName` nelle specifiche
- Annotazioni:
 - `trident.netapp.io/importOriginalName`: Nome volume sul backend
 - `trident.netapp.io/importBackendUUID`: UUID del backend in cui esiste il volume
 - `trident.netapp.io/notManaged` (*Facoltativo*): Impostare su `"true"` per i volumi non gestiti. L'impostazione predefinita è `"false"`.

Di seguito è riportato un esempio di specifica per l'importazione di un volume gestito:

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

2. Applica il file PVC YAML al tuo cluster Kubernetes:

```
kubectl apply -f <pvc-file>.yaml
```

Trident importerà automaticamente il volume e lo assocerà al PVC.

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 i `ontap-nas` e `ontap-nas-flexgroup` driver.



- Il `ontap-nas-economy` driver gestisce lo storage a livello di `qtree`, quindi non supporta l'importazione di un volume FlexVol autonomamente. Per importare `qtree` con il `ontap-nas-economy` driver, consulta [Importa qtree](#) (Tech Preview). Quando Trident importa un `qtree`, importa anche il FlexVol genitore come bucket che contiene il `qtree`.
- I `ontap-nas` e `ontap-nas-flexgroup` driver non consentono nomi di volumi duplicati.

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

Esempi di ONTAP NAS utilizzando `tridentctl`

Gli esempi seguenti mostrano come importare volumi gestiti e non gestiti utilizzando `tridentctl`.

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
pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7	1.0 GiB	standard
file	c5a6f6a4-b052-423b-80d4-8fb491a14a22	online

Volume non gestito

Quando si utilizza l'`--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      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

Esempi ONTAP NAS che utilizzano annotazioni PVC

I seguenti esempi mostrano come importare volumi gestiti e non gestiti utilizzando annotazioni PVC.

Volume gestito

Il seguente esempio importa un volume da 1GiB ontap-nas denominato `ontap_volume1` dal backend `81abcb27-ea63-49bb-b606-0a5315ac5f21` con modalità di accesso RWO impostata tramite annotazioni PVC:

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>

```

Volume non gestito

Il seguente esempio importa 1Gi ontap-nas volume denominato `ontap-volume2` dal backend `34abcb27-ea63-49bb-b606-0a5315ac5f34` con modalità di accesso RWO impostata tramite annotazioni PVC:

```

kind: PersistentVolumeClaim

```

```

apiVersion: v1
metadata:
  name: <unmanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>

```

Importa qtree



L'importazione qtree è una funzionalità Tech Preview in Trident 26.06. Le funzionalità Tech Preview non sono supportate per l'utilizzo in produzione.

A partire da Trident 26.06, puoi importare i qtree ONTAP esistenti in Trident usando il `ontap-nas-economy` driver. Identifica il qtree tramite i nomi del volume FlexVol e del qtree nel formato `flexvol/qtree`, insieme all'UUID del backend che lo contiene. Le considerazioni generali sull'importazione in [Panoramica e considerazioni](#) si applicano anche all'importazione dei qtree.

Importa un qtree usando `tridentctl`

Specifica il qtree nel formato `flexvol/qtree` come nome del volume. Il seguente esempio importa il qtree `qtreet1` dal volume FlexVol `flexvol1` sul backend `ontap_nas_economy`:

```
tridentctl import volume ontap_nas_economy flexvol1/qtreet1 -f <path-to-pvc-file>
```

Per importare il qtree senza rinominarlo, aggiungi il `--no-rename` flag. Per importare il qtree senza che Trident ne gestisca il ciclo di vita, aggiungi il `--no-manage` flag.

Importa un qtree usando le annotazioni PVC

Crea un PVC con le seguenti annotazioni:

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>

```

```

annotations:
  trident.netapp.io/importOriginalName: "flexvoll1/qtree1"
  trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>

```

Per mantenere i nomi esistenti del volume FlexVol e del qtree, aggiungi l' `trident.netapp.io/importNoRename: "true"` annotazione. Per importare il qtree come non gestito, aggiungi l' `trident.netapp.io/notManaged: "true"` annotazione.

Denominazione e comportamento gestionale

Importazione gestita (impostazione predefinita)

Trident rinomina il qtree in modo che corrisponda al nome del PVC e rinomina il FlexVol volume secondo la `ontap-nas-economy` convenzione di naming. Il percorso di giunzione non cambia.

--no-rename

Trident mantiene i nomi esistenti dei FlexVol volume e dei qtree. I nomi devono già seguire le convenzioni di naming di Trident, altrimenti Trident non può gestire le operazioni di eliminazione o montaggio del qtree.

--no-manage

Trident fa riferimento al qtree ma non ne gestisce il ciclo di vita. I nomi devono seguire le convenzioni di naming di Trident e Trident non utilizza né gestisce policy di esportazione automatica per il qtree.

Comportamento della policy di esportazione

Se il backend utilizza le policy di esportazione automatica, il qtree importato utilizza il modello di policy di esportazione per qtree, in cui ogni qtree ha la propria policy di esportazione. Trident assegna al qtree una policy di esportazione vuota al momento dell'importazione. Al momento della pubblicazione, Trident aggiorna il volume FlexVol alla policy di esportazione per backend se l'esportazione automatica è abilitata e la policy di esportazione FlexVol esistente è `default` o vuota.



L'aggiornamento della policy di esportazione del volume FlexVol può interrompere altri qtree nello stesso volume FlexVol che sono attualmente montati. Importa i qtree solo quando i qtree nel volume FlexVol sono smontati.



Non indirizzare due cluster Kubernetes allo stesso SVM attivo e non importare un qtree che un altro cluster gestisce ancora. Se entrambi i cluster gestiscono lo stesso volume, operazioni in conflitto possono corrompere o distruggere i dati.

ONTAP SAN

Trident supporta l'importazione di volumi utilizzando i `ontap-san` (iSCSI, NVMe/TCP e FC) e `ontap-san-economy` driver.

Trident può importare volumi ONTAP SAN FlexVol che contengono una singola LUN. Questo è coerente con il

driver `ontap-san`, che crea un volume FlexVol per ogni PVC e una LUN all'interno del volume FlexVol. Trident importa il volume FlexVol e lo associa alla definizione del PVC. Trident può importare volumi `ontap-san-economy` che contengono più LUN.

I seguenti esempi mostrano come importare volumi gestiti e non gestiti:

Volume gestito

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

Il seguente esempio 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 LUN mappate su igroups che condividono un IQN con un IQN di un nodo Kubernetes, come mostrato

nell'esempio seguente, riceverai l'errore: LUN already mapped to initiator(s) in this group. Dovrai rimuovere l'initiator o annullare la mappatura della LUN per importare il volume.

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

Esempi di ONTAP SAN-economy

I seguenti esempi mostrano come importare volumi gestiti e non gestiti per il backend. `ontap-san-economy`

Volume gestito

Quando si importa un volume gestito, Trident assume la proprietà del FlexVol e lo rinomina. È necessario tenere conto di questa ridenominazione quando si importano più LUN dallo stesso FlexVol.

Il seguente esempio importa `lun1` dal FlexVol `toimport` come volume gestito denominato `vol-managed-saneco`:

```
tridentctl import volume vol-managed-saneco toimport/lun1 -f
import1.yaml
```

Dopo l'importazione `lun1`, Trident rinomina il FlexVol (ad esempio, in `trident_lun_pool_xyz`). Per importare ulteriori LUN dallo stesso FlexVol, utilizzare il nuovo nome FlexVol:

```
tridentctl import volume vol-managed-saneco trident_lun_pool_xyz/lun2
-f import2.yaml
```



Il `ontap-san-economy` backend importa un LUN alla volta. Puoi automatizzare più importazioni utilizzando uno script.

Volume non gestito

Quando si importa un volume non gestito, Trident non assume la proprietà del FlexVol. Tuttavia, il FlexVol e la LUN devono seguire le convenzioni di denominazione di Trident.

Formato di denominazione FlexVol

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- STORAGEPREFIX è il valore di `storagePrefix` nella configurazione del backend. Il valore predefinito è `trident`.
- RANDOMSTRING è una qualsiasi stringa tu scelga.

Requisito di denominazione LUN

Il LUN deve essere denominato `lun0`.

Esempio

Se il tuo `storagePrefix` è `xyz`, il full path al LUN è:

```
trident_lun_pool_xyz_randomstring/lun0
```

Elemento

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



Il driver Element supporta nomi di volume duplicati. Tuttavia, Trident restituisce un errore se ci sono nomi di volume duplicati. Come soluzione alternativa, clona il volume, fornisci un nome di volume univoco e importa il volume clonato.

Il seguente esempio 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   |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

Azure NetApp Files

Trident supporta l'importazione di volumi utilizzando il `azure-netapp-files` driver.



Per importare un volume di Azure NetApp Files, identifica il volume tramite il suo percorso volume. Il percorso 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 volume è importvol1.

Il seguente esempio importa un azure-netapp-files volume sul backend azurenetaappfiles_40517 con il percorso del volume importvol1.

```
tridentctl import volume azurenetaappfiles_40517 importvol1 -f <path-to-pvc-file> -n trident
```

	NAME		SIZE		STORAGE CLASS
PROTOCOL	BACKEND UUID			STATE	
	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 il google-cloud-netapp-volumes driver.

Il seguente esempio importa un volume su 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	
	pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0		10 GiB		gcnv-nfs-sc-
identity	file	8c18cdf1-0770-4bc0-bcc5-c6295fe6d837		online	true

```
+-----+-----+
```

Il seguente esempio importa un `google-cloud-netapp-volumes` volume quando sono presenti due volumi 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 di Trident

Con Trident, puoi assegnare nomi ed etichette significativi ai volumi che crei. Questo ti aiuta a identificare e mappare facilmente i volumi alle rispettive risorse Kubernetes (PVC). Puoi anche definire modelli a livello di backend per creare nomi di volumi personalizzati ed etichette personalizzate; tutti i volumi che crei, importi o cloni aderiranno ai modelli.

Prima di iniziare

Supporto per nomi ed etichette di volume personalizzabili:

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

Limitazioni

- I nomi dei volumi personalizzati sono compatibili solo con i driver ONTAP on-premises.
- Le etichette personalizzate sono supportate solo per i `ontap-san`, `ontap-nas` e `ontap-nas-`

flexgroup driver.

- I nomi dei volumi personalizzati non si applicano ai volumi esistenti.

Comportamenti chiave dei nomi di volume personalizzabili

- 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 nominato secondo la convenzione di naming esistente.
- Il prefisso di archiviazione non è applicabile quando un volume viene nominato utilizzando un name template dalla configurazione del backend. Qualsiasi valore di prefisso desiderato può essere aggiunto direttamente al template.

Esempi di configurazione del backend con name template ed etichette

I modelli di nome personalizzati possono essere definiti a livello di root e/o 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.RequestName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

Esempio a livello di pool

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
```

```

"useREST": true,
"storage": [
  {
    "labels": {
      "labelname": "label1",
      "name": "{{ .volume.Name }}"
    },
    "defaults": {
      "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster }}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
    }
  },
  {
    "labels": {
      "cluster": "label2",
      "name": "{{ .volume.Name }}"
    },
    "defaults": {
      "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster }}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
    }
  }
]
}

```

Esempi di template di nome

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 importazione di volumi, le etichette vengono aggiornate solo se il volume esistente ha etichette in un formato specifico. Ad 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 di root nella definizione del backend.

3. Trident non supporta l'uso di un operatore slice con il prefisso dello storage.
4. Se i modelli non producono nomi di volume univoci, Trident aggiungerà alcuni caratteri casuali per creare nomi di volume univoci.
5. Se il nome personalizzato per un volume NAS economy supera i 64 caratteri, Trident nominerà i volumi in base alla convenzione di naming esistente. Per tutti gli altri driver ONTAP, se il nome del volume supera il limite di nome, il processo di creazione del volume non riesce.

Condividi un volume NFS Trident tra namespace

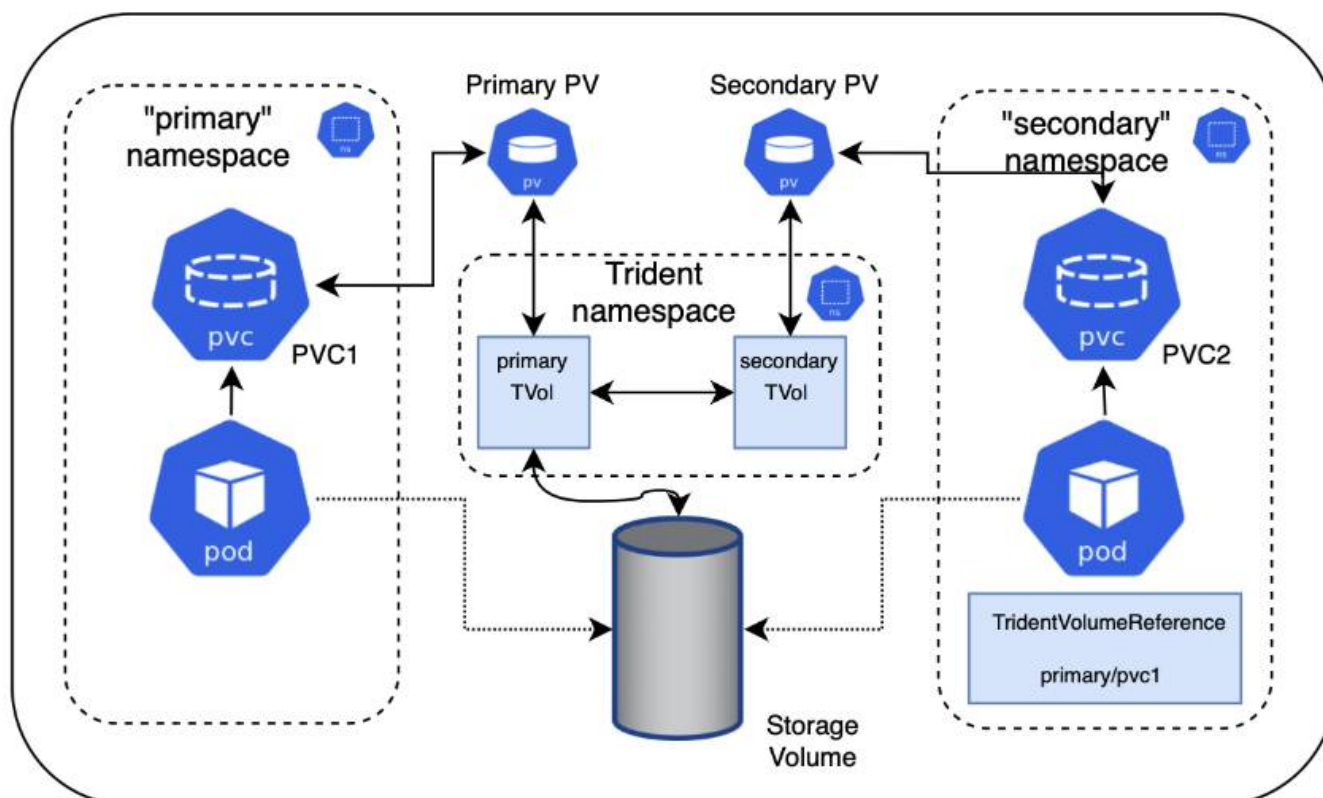
Utilizzando Trident, puoi creare un volume in uno spazio dei nomi primario e condividerlo in uno o più spazi dei nomi secondari.

Caratteristiche

La CR `TridentVolumeReference` consente di condividere in modo sicuro volumi NFS ReadWriteMany (RWX) su uno o più namespace Kubernetes. Questa soluzione nativa per 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 il permesso di accedere ai dati nel source PVC.

2

Concedere il permesso di creare un CR nello spazio dei nomi di destinazione

L'amministratore del cluster concede il permesso al proprietario dello spazio dei nomi di destinazione di creare il CR `TridentVolumeReference`.

3

Creare `TridentVolumeReference` nello spazio dei nomi di destinazione

Il proprietario dello spazio dei nomi di destinazione crea il `TridentVolumeReference` CR 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 del namespace di origine, dell'amministratore del cluster e del proprietario del namespace di destinazione. Il ruolo dell'utente viene assegnato in ogni fase.

Passaggi

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 l'annotazione `shareToNamespace`.

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

```
storage: 100Gi
```

Trident crea il PV e il suo volume di archiviazione NFS backend.



- È possibile condividere il PVC con più namespace utilizzando un elenco delimitato da virgole. Ad esempio, `trident.netapp.io/shareToNamespace: namespace2, namespace3, namespace4`.
- Puoi condividere con tutti gli spazi dei nomi utilizzando `*`. Ad esempio, `trident.netapp.io/shareToNamespace: *`
- È possibile aggiornare il PVC per includere l'annotazione `shareToNamespace` 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 di creare il CR `TridentVolumeReference` nello spazio dei nomi di destinazione.
3. **Proprietario dello spazio dei nomi di destinazione:** Creare un `TridentVolumeReference` CR nello spazio dei nomi di destinazione che faccia 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 l'annotazione `shareFromPVC` per designare il PVC di origine.

```
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

Trident legge l' ``shareFromPVC`` annotazione sul PVC di destinazione e crea il PV di destinazione come volume subordinato, senza risorse di storage proprie, che punta al PV di origine e condivide la risorsa di storage del PV di origine. Il PVC di destinazione e il PV di destinazione appaiono vincolati come di consueto.

Elimina un volume condiviso

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

Utilizzare `tridentctl get` per interrogare i volumi subordinati

Utilizzando l'[`tridentctl`] utility, puoi eseguire il ``get`` comando per ottenere i volumi subordinati. Per ulteriori informazioni, consultare ``tridentctl`` comandi e opzioni.

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

Flag:

- ``-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 ai namespace di destinazione di scrivere sul volume condiviso. È consigliabile utilizzare il blocco dei file o altri processi per impedire la sovrascrittura dei dati del volume condiviso.
- Non è possibile revocare l'accesso al PVC sorgente rimuovendo le `shareToNamespace` o `shareFromNamespace` annotazioni o eliminando il `TridentVolumeReference` CR. Per revocare l'accesso, è necessario eliminare il PVC subordinato.
- Snapshot, cloni e mirroring non sono possibili sui volumi subordinati.

Per ulteriori informazioni

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

- Guarda la demo su ["NetAppTV"](#).

Clona i volumi Trident tra namespace

Utilizzando Trident, è possibile creare nuovi volumi utilizzando volumi esistenti o `volumesnapshots` da un namespace 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 storage.



La clonazione tra spazi dei nomi è supportata solo per i driver di archiviazione `ontap-san` e `ontap-nas`. Le clonazioni di sola lettura non sono supportate.

Avvio rapido

È possibile configurare la clonazione dei volumi in pochi passaggi.

1

Configura il PVC sorgente per clonare il volume

Il proprietario dello spazio dei nomi di origine concede il permesso di accedere ai dati nel source PVC.

2

Concedere il permesso di creare un CR nello spazio dei nomi di destinazione

L'amministratore del cluster concede il permesso al proprietario dello spazio dei nomi di destinazione di creare il CR `TridentVolumeReference`.

3

Creare `TridentVolumeReference` nello spazio dei nomi di destinazione

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

4

Creare il PVC clone nello spazio dei nomi di destinazione

Il proprietario dello spazio dei nomi di destinazione crea un 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 spazi dei nomi 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 assegnato in ogni fase.

Passaggi

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

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
```

```

annotations:
  trident.netapp.io/cloneToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi

```

Trident crea il PV e il suo volume di storage backend.



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

2. **Cluster admin:** Assicurarsi che sia presente un RBAC appropriato per concedere il permesso al proprietario dello spazio dei nomi destinazione di creare il `TridentVolumeReference` CR nello spazio dei nomi destinazione (`namespace2`).
3. **Proprietario dello spazio dei nomi di destinazione:** Crea un CR `TridentVolumeReference` nello spazio dei nomi di destinazione che faccia riferimento al PVC 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:** Creare un PVC (`pvc2`) nello spazio dei nomi di destinazione (`namespace2`) usando le `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 provisionati utilizzando i driver ontap-nas-economy, i cloni di sola lettura non sono supportati.

Replicare i volumi utilizzando SnapMirror

Trident supporta le relazioni di mirroring tra un volume di origine su un cluster e il volume di destinazione sul cluster peered per replicare i dati per il disaster recovery. Puoi utilizzare una Custom Resource Definition (CRD) namespaced, chiamata Trident Mirror Relationship (TMR), per eseguire le seguenti operazioni:

- Crea relazioni di mirroring tra volumi (PVC)
- Rimuovere le relazioni di mirroring tra i volumi
- Interrompere le relazioni a specchio
- Promuovere il volume secondario durante le condizioni di disastro (failover)
- Esegui la transizione senza perdite delle applicazioni da un cluster a un altro cluster (durante i failover o le migrazioni pianificate)

Prerequisiti della replica

Assicurarsi che siano soddisfatti i seguenti prerequisiti prima di iniziare:

Cluster ONTAP

- **Trident:** Trident versione 22.10 o successiva deve essere presente sia sul cluster Kubernetes di origine che su quello 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. Consulta ["Panoramica delle licenze SnapMirror in ONTAP"](#) per maggiori informazioni.

A partire da ONTAP 9.10.1, tutte le licenze vengono fornite come NetApp license file (NLF), ovvero un singolo file che abilita più funzionalità. Per ulteriori informazioni, consulta ["Licenze incluse con ONTAP One"](#).



È supportata solo la protezione asincrona SnapMirror.

Peering

- **Cluster e SVM:** I backend di storage ONTAP devono essere sottoposti a peering. Consulta ["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 del volume con NetApp SnapMirror technology utilizzando classi di archiviazione supportate dai seguenti driver: **ontap-nas: NFS** **ontap-san: iSCSI** **ontap-san: FC** **ontap-san: NVMe/TCP** (richiede la versione di ONTAP minima 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 i sistemi di storage ASA r2"](#).

Crea un PVC specchiato

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

Passaggi

1. Eseguire i seguenti passaggi sul cluster Kubernetes primario:
 - a. Crea un oggetto StorageClass con il `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. Crea un PVC con la StorageClass creata in precedenza.

Esempio

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
    - ReadWriteMany
```

```
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

- c. Crea un MirrorRelationship CR 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 del MirrorRelationship.

- d. Ottieni il TridentMirrorRelationship CR 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. Crea un 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. Crea un MirrorRelationship CR con informazioni sulla destinazione e sulla sorgente.

Esempio

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  replicationPolicy: MirrorAllSnapshots
  replicationSchedule: 5min
  volumeMappings:
    - localPVCName: csi-nas
      remoteVolumeHandle:
        "datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Il TMR spec ha quattro campi di primo livello: `state`, che imposta lo stato di relazione desiderato (vedi [Stati di replicazione del volume](#)); `volumeMappings`, che associa i PVC locali ai relativi handle di volume remoti; e i campi facoltativi `replicationPolicy` e `replicationSchedule` descritti di seguito.

I campi `replicationPolicy` e `replicationSchedule` sono facoltativi. Imposta `replicationPolicy` sul nome di un criterio asincrono di SnapMirror su ONTAP e `replicationSchedule` sul nome di una pianificazione ONTAP. Se ometti uno dei due campi, Trident utilizza il criterio di replica o la pianificazione corrispondente configurata sul backend. Imposta questi campi quando crei la relazione; non puoi modificare il criterio o la pianificazione dopo che la relazione è stata stabilita. Per utilizzare impostazioni diverse, elimina e ricrea il TMR.

Trident crea e inizializza una relazione di SnapMirror utilizzando la policy di replica e la pianificazione configurate.

1. Creare un PVC con la StorageClass precedentemente creata per fungere da secondario (SnapMirror destinazione).

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 la presenza del TridentMirrorRelationship CRD 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 in peering con la SVM remota definita nel MirrorRelationship.

Stati di replicazione del volume

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

- **Stabilito:** il PVC locale è il volume di destinazione di una relazione mirror e questa è una nuova relazione.
- **Promosso:** il PVC locale è ReadWrite e montabile, con nessuna relazione speculare attualmente in vigore.
- **Ristabilito:** il PVC locale è il volume di destinazione di una relazione di SnapMirror ed era anche precedentemente in quella relazione di SnapMirror.
 - 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 una 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 il PVC secondario durante un failover pianificato

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

Passaggi

1. Sul cluster Kubernetes primario, crea uno snapshot del PVC e attendi fino a quando lo snapshot viene creato.
2. Sul cluster Kubernetes primario, crea il TridentTridentSnapshotInfoCR per ottenere i dettagli interni.

Esempio

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

- a. Sul cluster Kubernetes secondario, aggiorna il campo *spec.state* del CR *TridentMirrorRelationship* a *promoted* e *spec.volumeMappings[].promotedSnapshotHandle* al *internalName* dello snapshot.
3. Nel cluster Kubernetes secondario, confermare lo stato (campo *status.state*) di *TridentMirrorRelationship* su *promoted*.

Ripristina una relazione mirror dopo un failover

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

Passaggi

1. Nel cluster Kubernetes secondario, assicurati che i valori per il campo *spec.remoteVolumeHandle* su *TridentMirrorRelationship* siano aggiornati.
2. Sul cluster Kubernetes secondario, aggiorna il campo *spec.state* di *TridentMirrorRelationship* in *reestablished*.

Operazioni aggiuntive

Trident supporta le seguenti operazioni sui volumi primario e secondario:

Replica il PVC primario in un nuovo PVC secondario

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

Passaggi

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

Ridimensiona un PVC mirrorato, primario o secondario

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

Rimuovi la replicazione da un PVC

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

- Eliminare il *MirrorRelationship* sul PVC secondario. Questo interrompe la relazione di replicazione.

- Oppure, aggiorna il campo `spec.state` su *promoted*.

Elimina un PVC (che era stato precedentemente mirrorato)

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 mirror fa sì che il TMR rimanente passi allo stato *promoted* prima che Trident completi l'eliminazione. Se il TMR selezionato per l'eliminazione è già nello stato *promoted*, 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 questo volume verrà utilizzato in una relazione mirror in futuro, dovrà utilizzare un nuovo TMR con uno stato di replica del volume *established* durante la creazione della nuova relazione mirror.

Aggiorna le relazioni mirror quando ONTAP è online

Le relazioni mirror possono essere aggiornate in qualsiasi momento dopo essere state stabilite. È possibile utilizzare i campi `state: promoted` o `state: reestablished` per aggiornare le relazioni. Quando si promuove un volume di destinazione a un volume *ReadWrite* normale, è possibile utilizzare *promotedSnapshotHandle* per specificare uno snapshot specifico a 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 `TridentActionMirrorUpdate` CRD. Può assumere un valore tra *Succeeded*, *In Progress* o *Failed*.

Usa la topologia CSI

Trident può creare e collegare selettivamente i volumi ai nodi presenti in un cluster Kubernetes facendo uso del ["Funzione Topology 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. I provider di cloud oggi consentono agli amministratori di

Kubernetes di creare nodi basati sulle zone. I nodi possono essere situati in diverse zone di disponibilità all'interno di una regione o in varie regioni. Per facilitare il provisioning dei volumi per i carichi di lavoro in un'architettura multizona, Trident utilizza CSI Topology.



Scopri di più sulla funzione CSI Topology ["qui"](#).

Kubernetes offre due modalità uniche di binding dei volumi:

- Con `VolumeBindingMode` impostato su `Immediate`, Trident crea il volume senza alcuna consapevolezza della topologia. Il binding del volume e il provisioning dinamico vengono gestiti quando viene creato il PVC. Questo è il valore predefinito `VolumeBindingMode` ed è adatto per i cluster che non applicano vincoli topologici. I volumi persistenti vengono creati senza alcuna dipendenza dai requisiti di pianificazione del pod richiedente.
- Con `VolumeBindingMode` impostato su `WaitForFirstConsumer`, la creazione e il binding di un Persistent Volume per un PVC vengono ritardati fino a quando un pod che utilizza il PVC viene pianificato e creato. In questo modo, i volumi vengono creati per soddisfare i vincoli di pianificazione imposti dai requisiti topologici.



La `WaitForFirstConsumer` modalità di binding non richiede etichette di topologia. Questo può essere utilizzato indipendentemente dalla funzione Topologia CSI.

Cosa ti servirà

Per utilizzare la Topologia CSI, è necessario quanto segue:

- Un cluster Kubernetes che esegue un ["versione supportata di Kubernetes"](#)

```
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 del 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 che Trident sia installato affinché Trident sia consapevole della topologia.

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{ "\n" }{end}' | grep --color "topology.kubernetes.io"
[nodel,
{"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"}}]

```

Passo 1: Creare un backend consapevole della topologia

I backend di storage Trident possono essere progettati per fornire selettivamente volumi in base alle zone di disponibilità. Ogni backend può contenere un blocco opzionale `supportedTopologies` che rappresenta un elenco di zone e regioni supportate. Per le StorageClasses che fanno uso di un backend di questo tipo, un volume viene creato solo se richiesto da un'applicazione pianificata in una regione/zona supportata.

Ecco un esempio di definizione backend:

YAML

```

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

```

JSON

```

{
  "version": 1,

```

```

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

```



supportedTopologies è usato 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 un StorageClass. Per StorageClasses che contengono un sottoinsieme delle regioni e zone fornite in un backend, Trident crea un volume sul backend.

È possibile definire supportedTopologies per pool di storage anche. Vedere il seguente esempio:

```

---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-centrall1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-centrall1
    topology.kubernetes.io/zone: us-centrall1-a
  - topology.kubernetes.io/region: us-centrall1
    topology.kubernetes.io/zone: us-centrall1-b
storage:
  - labels:
      workload: production
    supportedTopologies:
      - topology.kubernetes.io/region: us-centrall1
        topology.kubernetes.io/zone: us-centrall1-a
    - labels:

```

```
workload: dev
supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
```

In questo esempio, le etichette `region` e `zone` indicano la posizione del pool di storage. `topology.kubernetes.io/region` e `topology.kubernetes.io/zone` stabiliscono da dove possono essere consumati i pool di storage.

Passo 2: Definire StorageClasses che sono consapevoli della topologia

In base alle etichette topologiche fornite ai nodi nel cluster, StorageClasses può essere definito per contenere informazioni sulla topologia. Questo determinerà i pool di storage che servono come candidati per le richieste di 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:
  name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions:
    - key: topology.kubernetes.io/zone
      values:
        - us-east1-a
        - us-east1-b
    - key: topology.kubernetes.io/region
      values:
        - us-east1
parameters:
  fsType: ext4
```

Nella definizione di StorageClass fornita sopra, `volumeBindingMode` è impostato su `WaitForFirstConsumer`. I PVC richiesti con questo StorageClass non saranno utilizzati finché non saranno referenziati in un pod. E, `allowedTopologies` fornisce le zone e la regione da utilizzare. Il `netapp-san-us-east1` StorageClass crea i PVC sul `san-backend-us-east1` backend definito sopra.

Fase 3: Creare e utilizzare un PVC

Con la StorageClass creata e mappata su un backend, ora puoi creare i PVC.

Vedi l'esempio spec qui 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 manifest avrebbe il seguente risultato:

```

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 consentire a Trident di creare un volume e associarlo al PVC, utilizzare il PVC in un pod. 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: voll
      persistentVolumeClaim:
        claimName: pvc-san
  containers:
    - name: sec-ctx-demo
      image: busybox
      command: [ "sh", "-c", "sleep 1h" ]
      volumeMounts:
        - name: voll
          mountPath: /data/demo
      securityContext:
        allowPrivilegeEscalation: false

```

Questo podSpec indica a Kubernetes di programmare il pod sui nodi presenti nella us-east1 regione, e di scegliere tra qualsiasi nodo presente nella us-east1-a o us-east1-b zone.

Vedi 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` utilizzando `tridentctl backend update`. Questo non influirà sui volumi che sono già stati forniti e sarà utilizzato solo per i PVC successivi.

Trova ulteriori informazioni

- ["Gestisci le risorse per i contenitori"](#)
- ["nodeSelector"](#)
- ["Affinità e anti-affinità"](#)
- ["Taints e Tolleranze"](#)

Lavora con gli snapshot di Trident

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 dai `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files` e `google-cloud-netapp-volumes` driver.

Prima di iniziare

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

Se la tua distribuzione Kubernetes non include il controller snapshot e i CRD, consulta [Distribuire un controller 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 integrato e nascosto.

Crea uno Snapshot del volume

Passaggi

1. Crea un `VolumeSnapshotClass`. Per ulteriori informazioni, fare riferimento a "[VolumeSnapshotClass](#)".
 - Il driver punta al driver Trident CSI.
 - `deletionPolicy` può essere `Delete` o `Retain`. Quando impostato su `Retain`, lo snapshot fisico sottostante sullo storage cluster viene mantenuto anche quando l'oggetto `VolumeSnapshot` 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 una Snapshot di un PVC esistente.

Esempi

- Questo esempio crea uno snapshot 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
```

- In questo esempio viene creato 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 oggetto `VolumeSnapshotContent` che rappresenta lo snapshot effettivo.

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

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

- È possibile identificare l'oggetto `VolumeSnapshotContent` per il `pvcl-snap` `VolumeSnapshot` descrivendolo. L'oggetto `Snapshot Content Name` identifica l'oggetto `VolumeSnapshotContent` che serve questa snapshot. Il parametro `Ready To Use` indica che la snapshot può essere utilizzata per creare un nuovo PVC.

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

Crea un PVC da una Snapshot del volume

Puoi usare `dataSource` per creare un PVC usando un `VolumeSnapshot` denominato `<pvc-name>` come origine dei dati. Dopo che il PVC è stato creato, può essere collegato a un pod e usato come qualsiasi altro PVC.



Il PVC verrà creato nello stesso backend del volume sorgente. Fai riferimento a ["KB: Creazione di un PVC da un Trident PVC Snapshot non può essere creata in un backend alternativo"](#).

L'esempio seguente crea il PVC utilizzando `pvcl-snap` come origine 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 una Snapshot del volume

Trident supporta la "[Processo di snapshot pre-provisionato Kubernetes](#)" per consentire all'amministratore del cluster di creare un oggetto `VolumeSnapshotContent` e importare Snapshot creati al di fuori di Trident.

Prima di iniziare

Trident deve aver creato o importato il volume d'origine dello Snapshot.

Passaggi

1. **Cluster admin:** Creare un `VolumeSnapshotContent` oggetto che fa riferimento allo snapshot del backend. Questo avvia il flusso di lavoro dello 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 nella chiamata `ListSnapshots`.



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

Esempio

Il seguente esempio crea un oggetto `VolumeSnapshotContent` 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. **Cluster admin:** Crea il VolumeSnapshot CR che fa riferimento all' VolumeSnapshotContent oggetto. Questa richiesta consente l'accesso all'utilizzo di VolumeSnapshot in un determinato namespace.

Esempio

Il seguente esempio crea un VolumeSnapshot CR chiamato import-snap che fa riferimento al VolumeSnapshotContent chiamato 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 VolumeSnapshotContent e esegue la chiamata ListSnapshots. Trident crea il TridentSnapshot.
 - L'external snapshotter imposta VolumeSnapshotContent su readyToUse e VolumeSnapshot su true.
 - Trident restituisce readyToUse=true.
4. **Qualsiasi utente:** Crea un PersistentVolumeClaim per fare riferimento al nuovo VolumeSnapshot, dove il spec.dataSource (o spec.dataSourceRef) nome è il VolumeSnapshot nome.

Esempio

Il seguente esempio crea un PVC che fa riferimento al VolumeSnapshot denominato 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 le Snapshot

La directory delle snapshot è nascosta per impostazione predefinita per facilitare la massima compatibilità dei volumi forniti utilizzando i driver `ontap-nas` e `ontap-nas-economy`. Abilita la directory `.snapshot` per recuperare i dati direttamente dalle snapshot.

Utilizzare il comando ONTAP CLI di ripristino dell'istantanea del volume per ripristinare un volume a uno stato registrato in una precedente Snapshot.

```
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 vengono perse.

Ripristino del volume in loco da una Snapshot

Trident offre un rapido ripristino in-place del volume da una snapshot utilizzando il `TridentActionSnapshotRestore` (TASR) CR. Questo CR funziona come un'azione Kubernetes imperativa e non persiste dopo il completamento dell'operazione.

Trident supporta il ripristino delle snapshot sui driver `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes` e `solidfire-san`.

Prima di iniziare

È necessario disporre di un PVC vincolato e di una Snapshot del volume disponibile.

- Verificare che lo stato del PVC sia `bound`.

```
kubectl get pvc
```

- Verificare che la snapshot del volume sia pronta per l'uso.

```
kubectl get vs
```

Passaggi

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



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

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

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

Risultati

Trident ripristina i dati dallo snapshot. Puoi 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 fallimento. Dovrai eseguire nuovamente l'operazione.
- Gli utenti Kubernetes senza accesso admin potrebbero dover ottenere il permesso dall'admin per creare un TASR CR nel loro namespace dell'applicazione.

Eliminare un PV con le relative Snapshot

Quando si elimina un Persistent Volume con le relative Snapshot, il volume Trident corrispondente viene aggiornato a uno "Deleting state". Rimuovere le Snapshot del volume per eliminare il volume Trident.

Distribuire un controller snapshot del volume

Se la tua distribuzione di Kubernetes non include il controller di snapshot e i CRD, puoi distribuirli come segue.

Passaggi

1. Crea 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.yaml
1
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Crea il controller di 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, apri `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e aggiorna namespace al tuo namespace.

Link correlati

- ["Istantanee del volume"](#)
- ["VolumeSnapshotClass"](#)

Lavora con gli snapshot dei gruppi di volumi di Trident

Snapshot del gruppo di volumi Kubernetes dei 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 acquisite nello stesso point-in-time.



VolumeGroupSnapshot is a beta feature in Kubernetes con API beta. Kubernetes 1.32 è la versione minima richiesta per VolumeGroupSnapshot.

Crea snapshot del gruppo di volumi

Lo snapshot del gruppo di volumi è supportato con i seguenti driver di archiviazione:

- `ontap-san` driver - solo per i protocolli iSCSI e FC, non per il protocollo NVMe/TCP.
- `ontap-san-economy` - solo per il protocollo iSCSI.
- `ontap-nas`



Lo snapshot del gruppo di volumi non è supportato per i sistemi di archiviazione NetApp ASA r2 o AFX.

Prima di iniziare

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

Se la tua distribuzione Kubernetes non include il controller snapshot esterno e i CRD, consulta [Distribuire un controller snapshot del volume](#).



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

- Nel file YAML del controller snapshot, impostare il `CSIVolumeGroupSnapshot` feature gate su 'true' per garantire che la funzionalità di snapshot del gruppo di volumi sia abilitata.
- Crea le classi di snapshot del gruppo di volumi richieste prima di creare uno snapshot del gruppo di volumi.
- Assicurati che tutti i PVC/volumi siano sullo stesso SVM per poter creare `VolumeGroupSnapshot`.

Passaggi

- Crea un `VolumeGroupSnapshotClass` prima di creare un `VolumeGroupSnapshot`. Per ulteriori informazioni, consulta "[VolumeGroupSnapshotClass](#)".

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- Crea PVC con le etichette richieste utilizzando le classi di storage esistenti oppure aggiungi queste etichette ai PVC esistenti.

L'esempio seguente crea il PVC utilizzando `pvc1-group-snap` come origine 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` specificata nel PVC).

Questo esempio crea una Snapshot di un 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 una Snapshot di gruppo

È possibile ripristinare i singoli Persistent Volumes utilizzando le singole Snapshot che sono state create come parte del Volume Group Snapshot. Non è possibile recuperare il Volume Group Snapshot come unità.

Utilizzare il comando ONTAP CLI di ripristino dell'istantanea del volume per ripristinare un volume a uno stato registrato in una precedente Snapshot.

```

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 vengono perse.

Ripristino del volume in loco da una Snapshot

Trident offre un rapido ripristino in-place del volume da una snapshot utilizzando il `TridentActionSnapshotRestore` (TASR) CR. Questo CR funziona come un'azione Kubernetes imperativa e non persiste dopo il completamento dell'operazione.

Per ulteriori informazioni, consulta ["Ripristino del volume in loco da una Snapshot"](#).

Eliminare un PV con le Snapshot di gruppo associate

Quando si elimina una Snapshot di un volume di gruppo:

- È possibile eliminare VolumeGroupSnapshots come un tutto, non le singole snapshot nel gruppo.
- Se i PersistentVolumes vengono eliminati mentre esiste una snapshot per quel PersistentVolume, Trident sposterà quel volume in uno stato di "eliminazione" perché la snapshot deve essere rimossa prima che il volume possa essere rimosso in modo sicuro.
- Se è stato creato un clone utilizzando una snapshot raggruppata e poi il gruppo deve essere eliminato, inizierà un'operazione di split-on-clone e il gruppo non potrà essere eliminato fino al completamento della divisione.

Distribuire un controller snapshot del volume

Se la tua distribuzione di Kubernetes non include il controller di snapshot e i CRD, puoi distribuirli come segue.

Passaggi

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

2. Crea il controller di 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, apri `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e aggiorna namespace al tuo namespace.

Link correlati

- ["VolumeGroupSnapshotClass"](#)
- ["Istantanee del volume"](#)

Informazioni sul copyright

Copyright © 2026 NetApp, Inc. Tutti i diritti riservati. Stampato negli Stati Uniti d'America. Nessuna porzione di questo documento soggetta a copyright può essere riprodotta in qualsiasi formato o mezzo (grafico, elettronico o meccanico, inclusi fotocopie, registrazione, nastri o storage in un sistema elettronico) senza previo consenso scritto da parte del detentore del copyright.

Il software derivato dal materiale sottoposto a copyright di NetApp è soggetto alla seguente licenza e dichiarazione di non responsabilità:

IL PRESENTE SOFTWARE VIENE FORNITO DA NETAPP "COSÌ COM'È" E SENZA QUALSIVOGLIA TIPO DI GARANZIA IMPLICITA O ESPRESSA FRA CUI, A TITOLO ESEMPLIFICATIVO E NON ESAUSTIVO, GARANZIE IMPLICITE DI COMMERCIALIZZABILITÀ E IDONEITÀ PER UNO SCOPO SPECIFICO, CHE VENGONO DECLINATE DAL PRESENTE DOCUMENTO. NETAPP NON VERRÀ CONSIDERATA RESPONSABILE IN ALCUN CASO PER QUALSIVOGLIA DANNO DIRETTO, INDIRETTO, ACCIDENTALE, SPECIALE, ESEMPLARE E CONSEGUENZIALE (COMPRESI, A TITOLO ESEMPLIFICATIVO E NON ESAUSTIVO, PROCUREMENT O SOSTITUZIONE DI MERCI O SERVIZI, IMPOSSIBILITÀ DI UTILIZZO O PERDITA DI DATI O PROFITTI OPPURE INTERRUZIONE DELL'ATTIVITÀ AZIENDALE) CAUSATO IN QUALSIVOGLIA MODO O IN RELAZIONE A QUALUNQUE TEORIA DI RESPONSABILITÀ, SIA ESSA CONTRATTUALE, RIGOROSA O DOVUTA A INSOLVENZA (COMPRESA LA NEGLIGENZA O ALTRO) INSORTA IN QUALSIASI MODO ATTRAVERSO L'UTILIZZO DEL PRESENTE SOFTWARE ANCHE IN PRESENZA DI UN PREAVVISO CIRCA L'EVENTUALITÀ DI QUESTO TIPO DI DANNI.

NetApp si riserva il diritto di modificare in qualsiasi momento qualunque prodotto descritto nel presente documento senza fornire alcun preavviso. NetApp non si assume alcuna responsabilità circa l'utilizzo dei prodotti o materiali descritti nel presente documento, con l'eccezione di quanto concordato espressamente e per iscritto da NetApp. L'utilizzo o l'acquisto del presente prodotto non comporta il rilascio di una licenza nell'ambito di un qualche diritto di brevetto, marchio commerciale o altro diritto di proprietà intellettuale di NetApp.

Il prodotto descritto in questa guida può essere protetto da uno o più brevetti degli Stati Uniti, esteri o in attesa di approvazione.

LEGENDA PER I DIRITTI SOTTOPOSTI A LIMITAZIONE: l'utilizzo, la duplicazione o la divulgazione da parte degli enti governativi sono soggetti alle limitazioni indicate nel sottoparagrafo (b)(3) della clausola Rights in Technical Data and Computer Software del DFARS 252.227-7013 (FEB 2014) e FAR 52.227-19 (DIC 2007).

I dati contenuti nel presente documento riguardano un articolo commerciale (secondo la definizione data in FAR 2.101) e sono di proprietà di NetApp, Inc. Tutti i dati tecnici e il software NetApp forniti secondo i termini del presente Contratto sono articoli aventi natura commerciale, sviluppati con finanziamenti esclusivamente privati. Il governo statunitense ha una licenza irrevocabile limitata, non esclusiva, non trasferibile, non cedibile, mondiale, per l'utilizzo dei Dati esclusivamente in connessione con e a supporto di un contratto governativo statunitense in base al quale i Dati sono distribuiti. Con la sola esclusione di quanto indicato nel presente documento, i Dati non possono essere utilizzati, divulgati, riprodotti, modificati, visualizzati o mostrati senza la previa approvazione scritta di NetApp, Inc. I diritti di licenza del governo degli Stati Uniti per il Dipartimento della Difesa sono limitati ai diritti identificati nella clausola DFARS 252.227-7015(b) (FEB 2014).

Informazioni sul marchio commerciale

NETAPP, il logo NETAPP e i marchi elencati alla pagina <http://www.netapp.com/TM> sono marchi di NetApp, Inc. Gli altri nomi di aziende e prodotti potrebbero essere marchi dei rispettivi proprietari.