



Provisioning e gestione dei volumi

Astra Trident

NetApp

January 14, 2026

Sommario

Provisioning e gestione dei volumi	1
Provisioning di un volume	1
Panoramica	1
Creare PV e PVC	4
Espandere i volumi	5
Espandere un volume iSCSI	5
Espandere un volume NFS	10
Importa volumi	13
Panoramica e considerazioni	13
Importare un volume	14
Esempi	15
Personalizzare i nomi e le etichette dei volumi	20
Prima di iniziare	20
Limitazioni	20
Comportamenti chiave dei nomi di volume personalizzabili	20
Esempi di configurazione backend con modello di nome ed etichette	20
Esempi di modelli di nome	22
Punti da considerare	23
Condividere un volume NFS tra spazi dei nomi	23
Caratteristiche	23
Avvio rapido	24
Configurare gli spazi dei nomi di origine e di destinazione	25
Eliminare un volume condiviso	26
Utilizzare <code>tridentctl get</code> per eseguire query sui volumi subordinati	26
Limitazioni	27
Per ulteriori informazioni	27
Replica dei volumi con SnapMirror	27
Prerequisiti per la replica	27
Creare un PVC specchiato	28
Stati di replica dei volumi	31
Promozione del PVC secondario durante un failover non pianificato	31
Promozione del PVC secondario durante un failover pianificato	32
Ripristinare una relazione di mirroring dopo un failover	32
Operazioni supplementari	32
Aggiorna relazioni mirror quando ONTAP è online	33
Aggiorna relazioni di mirroring quando ONTAP non è in linea	33
Abilita Astra Control Provisioner	34
Utilizzare la topologia CSI	43
Panoramica	43
Fase 1: Creazione di un backend compatibile con la topologia	45
Fase 2: Definire StorageClasses che siano compatibili con la topologia	47
Fase 3: Creare e utilizzare un PVC	48

Aggiorna i backend da includere <code>supportedTopologies</code>	51
Trova ulteriori informazioni	51
Lavorare con le istantanee	51
Panoramica	51
Creare un'istantanea del volume	52
Creare un PVC da uno snapshot di volume	53
Importare uno snapshot di volume	54
Ripristinare i dati del volume utilizzando le snapshot	56
Ripristino del volume in-place da uno snapshot	56
Eliminare un PV con gli snapshot associati	58
Implementare un controller per lo snapshot dei volumi	58
Link correlati	59

Provisioning e gestione dei volumi

Provisioning di un volume

Creare un PersistentVolume (PV) e un PersistentVolumeClaim (PVC) che utilizza Kubernetes StorageClass configurato per richiedere l'accesso al PV. È quindi possibile montare il PV su un pod.

Panoramica

Un "[PersistentVolume](#)" (PV) è una risorsa di storage fisico fornita dall'amministratore del cluster in un cluster Kubernetes. Il "[PersistentVolumeClaim](#)" (PVC) è una richiesta di accesso al PersistentVolume sul cluster.

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

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

Manifesti campione

Manifesto di esempio di PersistentVolume

Questo manifesto di esempio mostra un PV di base di 10Gi associato a StorageClass basic-csi.

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

Manifesti di campioni PersistentVolumeClaim

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

PVC con accesso RWO

Questo esempio mostra un PVC di base con accesso RWO associato a 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 una classe StorageClass denominata protection-gold.

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

Campioni manifesti pod

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

Configurazione di base

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
        claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
  volumeMounts:
    - mountPath: "/my/mount/path"
      name: pv-storage
```

Configurazione NVMe/TCP di base

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

Creare PV e PVC

Fasi

1. Creare il PV.

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

2. Verificare lo stato FV.

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

3. Creare il PVC.

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

4. Verificare lo stato del PVC.

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

5. Montare il volume in un pod.

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



È possibile monitorare l'avanzamento utilizzando `kubectl get pod --watch`.

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

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

7. A questo punto è possibile eliminare il pod. L'applicazione Pod non esisterà più, ma il volume rimarrà.

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

Per ulteriori informazioni sull'interazione delle classi di storage con i `PersistentVolumeClaim` parametri e per il controllo del provisioning dei volumi da parte di Astra Trident, consulta "[Kubernetes e Trident Objects](#)".

Espandere i volumi

Astra Trident offre agli utenti Kubernetes la possibilità di espandere i propri volumi dopo la loro creazione. Informazioni sulle configurazioni richieste per espandere i volumi iSCSI e NFS.

Espandere un volume iSCSI

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



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

Fase 1: Configurare StorageClass per supportare l'espansione dei volumi

Modificare la definizione StorageClass per impostare il `allowVolumeExpansion` campo 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 in modo da includere il `allowVolumeExpansion` parametro.

Fase 2: Creare un PVC con la StorageClass creata

Modificare la definizione PVC e aggiornare `spec.resources.requests.storage` per riflettere le nuove dimensioni desiderate, che devono essere superiori alle dimensioni originali.

```
cat pvc-ontapsan.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Astra Trident crea un volume persistente (PV) e lo associa a questo PVC (Persistent Volume Claim).

```
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 colleghi il PVC

Collegare il PV a un pod affinché venga ridimensionato. Esistono due scenari quando si ridimensiona un PV iSCSI:

- Se il PV è collegato a un pod, Astra Trident espande il volume sul backend dello storage, esegue di nuovo la scansione del dispositivo e ridimensiona il file system.
- Quando si tenta di ridimensionare un PV non collegato, Astra Trident espande il volume sul backend dello storage. Dopo aver associato il PVC a un pod, Trident esegue nuovamente la scansione del dispositivo e ridimensiona il file system. Kubernetes aggiorna quindi le dimensioni del PVC dopo il completamento dell'operazione di espansione.

In questo esempio, viene creato un pod che utilizza 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 PVC e aggiornare spec.resources.requests.storage a 2Gi.

```

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

```

Fase 5: Convalidare l'espansione

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

```

kubectl get pvc san-pvc
NAME      STATUS      VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc    Bound      pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi
RWO          ontap-san   11m

kubectl get pv
NAME                                     CAPACITY      ACCESS MODES
RECLAIM POLICY  STATUS      CLAIM      STORAGECLASS  REASON      AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi          RWO
Delete        Bound      default/san-pvc  ontap-san      12m

tridentctl get volumes -n trident
+-----+-----+-----+
+-----+-----+-----+
|           NAME          |  SIZE   | STORAGE CLASS | 
PROTOCOL |           BACKEND UUID      | STATE   | MANAGED | 
+-----+-----+-----+
+-----+-----+-----+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san      | 
block     | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online  | true      | 
+-----+-----+-----+
+-----+-----+-----+

```

Espandere un volume NFS

Astra Trident supporta l'espansione del volume per NFS PVS con provisioning su ontap-nas, , ontap-nas-economy ontap-nas-flexgroup , gcp-cvs e azure-netapp-files backend.

Fase 1: Configurare StorageClass per supportare l'espansione dei volumi

Per ridimensionare un PV NFS, l'amministratore deve prima configurare la classe di archiviazione per consentire l'espansione del volume impostando il `allowVolumeExpansion` campo 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 è già stata creata una classe di archiviazione senza questa opzione, è possibile modificare semplicemente la classe di archiviazione esistente utilizzando `kubectl edit storageclass` per consentire l'espansione del volume.

Fase 2: Creare un PVC con la StorageClass creata

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

Astra Trident deve creare un PV NFS 20MiB 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
RECLAIM POLICY  STATUS    CLAIM           CAPACITY      ACCESS MODES
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi      RWO
Delete        Bound     default/ontapnas20mb  ontapnas
2m42s
```

Fase 3: Espandere il PV

Per ridimensionare il PV 20MiB appena creato a 1GiB, modificare il PVC e impostarlo `spec.resources.requests.storage` su 1GiB:

```

kubectl edit pvc ontapnas20mb
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
...

```

Fase 4: Convalidare l'espansione

È possibile verificare che il ridimensionamento funzioni correttamente controllando le dimensioni del volume PVC, PV e Astra 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
RECLAIM POLICY  STATUS  CLAIM
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      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

Importa volumi

È possibile importare i volumi di storage esistenti come Kubernetes PV utilizzando tridentctl import.

Panoramica e considerazioni

È possibile importare un volume in Astra Trident per:

- Containerizzare un'applicazione e riutilizzare il set di dati esistente
- Utilizzare un clone di un set di dati per un'applicazione temporanea
- Ricostruire un cluster Kubernetes guasto
- Migrazione dei dati delle applicazioni durante il disaster recovery

Considerazioni

Prima di importare un volume, esaminare le seguenti considerazioni.

- Astra Trident può importare solo volumi ONTAP di tipo RW (Read-write). I volumi di tipo DP (data Protection) sono volumi di destinazione SnapMirror. Prima di importare il volume in Astra Trident, è

necessario interrompere la relazione di mirroring.

- Si consiglia di importare volumi senza connessioni attive. Per importare un volume utilizzato attivamente, clonare il volume ed eseguire l'importazione.



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

- Sebbene `StorageClass` debba essere specificato su un PVC, Astra Trident non utilizza questo parametro durante l'importazione. Le classi di storage vengono utilizzate durante la creazione del volume per selezionare i pool disponibili in base alle caratteristiche dello storage. Poiché il volume esiste già, durante l'importazione non è richiesta alcuna selezione del pool. Pertanto, l'importazione non avrà esito negativo anche se il volume esiste in un backend o in un pool che non corrisponde alla classe di storage specificata nel PVC.
- La dimensione del volume esistente viene determinata e impostata nel PVC. Una volta importato il volume dal driver di storage, il PV viene creato con un `ClaimRef` sul PVC.
 - La politica di recupero viene inizialmente impostata su `retain` nel PV. Dopo che Kubernetes ha eseguito il binding con PVC e PV, la policy di recupero viene aggiornata in modo da corrispondere alla policy di recupero della classe di storage.
 - Se il criterio di recupero della classe di archiviazione è `delete`, il volume di archiviazione verrà eliminato quando il PV viene eliminato.
- Per impostazione predefinita, Astra Trident gestisce il PVC e rinomina il FlexVol e il LUN sul backend. È possibile passare il `--no-manage` flag per importare un volume non gestito. Se si utilizza `--no-manage`, Astra Trident non esegue alcuna operazione aggiuntiva sul PVC o sul PV per il ciclo di vita degli oggetti. Il volume di storage non viene cancellato quando il PV viene cancellato e vengono ignorate anche altre operazioni come il clone del volume e il ridimensionamento del volume.



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

- Al PVC e al PV viene aggiunta un'annotazione che serve a doppio scopo per 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 utilizzare `tridentctl import` per importare un volume.

Fasi

1. Creare il file PVC (Persistent Volume Claim) (ad esempio, `pvc.yaml`) che verrà utilizzato per creare il PVC. Il file PVC deve includere `name`, `namespace`, `accessModes` e `storageClassName`. Facoltativamente, è possibile specificare `unixPermissions` nella definizione del PVC.

Di seguito viene riportato un esempio di specifica minima:

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



Non includere parametri aggiuntivi come il nome PV o le dimensioni del volume. Questo può causare l'errore del comando di importazione.

2. Utilizza il `tridentctl import` comando per specificare il nome del back-end di Astra Trident che contiene il volume e il nome che identifica in modo univoco il volume sullo storage (ad esempio: ONTAP FlexVol, Element Volume, Cloud Volumes Service path). L' `-f` argomento è necessario per specificare il percorso del file PVC.

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

Esempi

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

NAS ONTAP e NAS FlexGroup ONTAP

Astra Trident supporta l'importazione di volumi utilizzando `ontap-nas` driver e `ontap-nas-flexgroup` .



- Il `ontap-nas-economy` driver non può importare e gestire le qtree.
- I `ontap-nas` driver e `ontap-nas-flexgroup` non consentono nomi di volumi duplicati.

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

Esempi NAS ONTAP

Di seguito viene illustrato un esempio di importazione di un volume gestito e di un volume non gestito.

Volume gestito

Nell'esempio seguente viene importato un volume denominato `managed_volume` in un backend denominato `ontap_nas`:

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

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

Volume non gestito

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

Nell'esempio seguente vengono importate `unmanaged_volume` sul `ontap_nas` backend:

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

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

ONTAP SAN

Astra Trident supporta l'importazione di volumi utilizzando il `ontap-san` driver. L'importazione di volumi non è supportata con il `ontap-san-economy` driver.

Astra Trident può importare SAN FlexVol ONTAP che contengono una singola LUN. Questa operazione è coerente con il `ontap-san` driver, che crea una FlexVol per ogni PVC e un LUN all'interno della FlexVol. Astra Trident importa il FlexVol e lo associa alla definizione del PVC.

Esempi DI SAN ONTAP

Di seguito viene illustrato un esempio di importazione di un volume gestito e di un volume non gestito.

Volume gestito

Per i volumi gestiti, Astra Trident rinomina FlexVol nel `pvc-<uuid>` formato e il LUN in `FlexVol` in `lun0`.

Nell'esempio seguente viene importato il `ontap-san-managed` `FlexVol` 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

Nell'esempio seguente vengono importate `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 ad igroup che condividono un IQN con un nodo Kubernetes IQN, come mostrato nell'esempio seguente, riceverai l'errore: `LUN already mapped to initiator(s) in this group.` Per importare il volume, è necessario rimuovere l'iniziatore o annullare la mappatura del LUN.

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

Elemento

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



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

Esempio di elemento

Nell'esempio seguente viene importato un `element-managed volume` sul backend `element_default`.

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

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

Piattaforma Google Cloud

Astra Trident supporta l'importazione di volumi utilizzando il `gcp-cvs` driver.



Per importare un volume supportato da NetApp Cloud Volumes Service in Google Cloud Platform, identificare il volume in base al relativo percorso. Il percorso del volume è la parte del percorso di esportazione del volume dopo `:`. Ad esempio, se il percorso di esportazione è `10.0.0.1:/adroit-jolly-swift`, il percorso del volume è `adroit-jolly-swift`.

Esempio di piattaforma Google Cloud

Nell'esempio seguente viene importato un `gcp-cvs` volume sul backend `gcpcvs_YEppr` con il percorso del volume di `adroit-jolly-swift`.

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

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

Azure NetApp Files

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



Per importare un volume Azure NetApp Files, identificare il volume in base al relativo percorso. Il percorso del volume è la parte del percorso di esportazione del volume dopo `:`. Ad esempio, se il percorso di montaggio è `10.0.0.2:/importvol1`, il percorso del volume è `importvol1`.

Esempio di Azure NetApp Files

Nell'esempio seguente viene importato un `azure-netapp-files` volume sul backend `azurenatappfiles_40517` con il percorso del volume `importvol1`.

```
tridentctl import volume azurenatappfiles_40517 importvol1 -f <path-to-pvc-file> -n trident

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

Personalizzare i nomi e le etichette dei volumi

Con Astra 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). È inoltre possibile definire modelli di backend per la creazione di nomi di volumi personalizzati ed etichette personalizzate; i volumi creati, importati o clonati aderiranno ai modelli.

Prima di iniziare

Nomi di volumi ed etichette personalizzabili supportano:

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

Limitazioni

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

Comportamenti chiave dei nomi di volume personalizzabili

1. Se si verifica un errore a causa di una sintassi non valida in un modello di nome, la creazione del backend non riesce. Tuttavia, se l'applicazione modello non riesce, il volume verrà denominato in base alla convenzione di denominazione esistente.
2. Il prefisso di archiviazione non è applicabile quando un volume viene nominato utilizzando un modello di nome dalla configurazione backend. Qualsiasi valore di prefisso desiderato può essere aggiunto direttamente al modello.

Esempi di configurazione backend con modello di nome ed etichette

I modelli con nomi personalizzati possono essere definiti a livello di root e/o pool.

Esempio di livello root

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

Esempio di livello 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 modelli 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 importazioni di volumi, le etichette vengono aggiornate solo se il volume esistente presenta 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 al livello principale nella definizione di backend.
3. Astra Trident non supporta l'utilizzo di un operatore slice con il prefisso di storage.
4. Se i modelli non danno come risultato nomi di volume unici, Astra Trident aggiungerà alcuni caratteri casuali per creare nomi di volume unici.
5. Se il nome personalizzato di un volume di economia NAS supera i 64 caratteri, Astra Trident denominerà i volumi in base alla convenzione di naming esistente. Per tutti gli altri driver ONTAP, se il nome del volume supera il limite del nome, il processo di creazione del volume non riesce.

Condividere un volume NFS tra spazi dei nomi

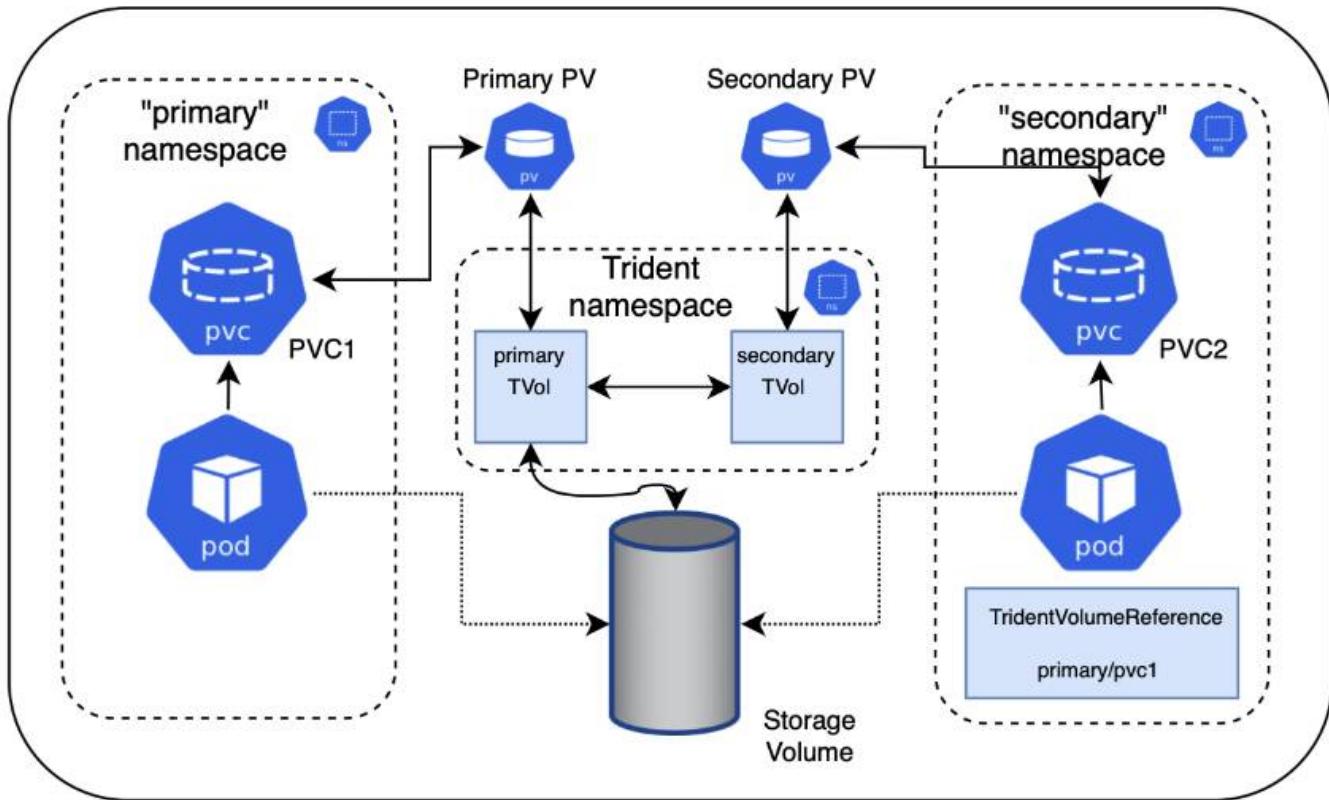
Utilizzando Astra Trident, è possibile creare un volume in uno spazio dei nomi primario e condividerlo in uno o più spazi dei nomi secondari.

Caratteristiche

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

- Diversi livelli di controllo degli accessi per garantire la sicurezza
- Funziona con tutti i driver di volume NFS Trident
- Nessuna dipendenza da tridentctl o da altre funzionalità Kubernetes non native

Questo diagramma illustra la condivisione del volume NFS tra due spazi dei nomi Kubernetes.



Avvio rapido

Puoi configurare la condivisione dei volumi NFS in pochi 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 PVC di origine.

2

Concedere l'autorizzazione per creare una CR nello spazio dei nomi di destinazione

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

3

Creare TridentVolumeReference nello spazio dei nomi di destinazione

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

4

Creare 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 di spazi dei nomi incrociati richiede la collaborazione e l'azione del proprietario dello spazio dei nomi di origine, dell'amministratore del cluster e del proprietario dello spazio dei nomi di destinazione. Il ruolo dell'utente viene designato in ogni fase.

Fasi

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

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

Astra Trident crea il PV e il suo volume di storage NFS back-end.

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



2. **Cluster admin:** creare il ruolo personalizzato e il kubeconfig per concedere l'autorizzazione al proprietario dello spazio dei nomi di destinazione per creare il CR di 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 fa riferimento allo spazio dei nomi di origine pvc1 .

```

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

```

4. **Proprietario dello spazio dei nomi di destinazione:** creare un PVC (pvc2) nello spazio dei nomi di destinazione ('namespace2' utilizzando l' `shareFromPVC` annotazione 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 al PVC di origine.

Risultati

Astra Trident legge l' `shareFromPVC` annotazione sul PVC di destinazione e crea il PV di destinazione come volume subordinato senza una risorsa di storage propria che punta al PV di origine e condivide la risorsa di storage PV di origine. Il PVC e il PV di destinazione appaiono associati come normali.

Eliminare un volume condiviso

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

Utilizzare `tridentctl get` per eseguire query sui volumi subordinati

Utilizzando l'[`tridentctl`` utilità, è possibile eseguire `get il comando per ottenere volumi subordinati. Per ulteriori informazioni, fare riferimento al [tridentctl Commands and options](#).

Usage:

```
tridentctl get [option]
```

Allarmi:

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

Limitazioni

- Astra Trident non può impedire la scrittura degli spazi dei nomi di destinazione nel volume condiviso. È necessario utilizzare il blocco dei file o altri processi per impedire la sovrascrittura dei dati dei volumi condivisi.
- Non è possibile revocare l'accesso al PVC di origine rimuovendo le shareToNamespace annotazioni o shareFromNamespace 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 ulteriori informazioni sull'accesso ai volumi tra spazi dei nomi:

- Visita "[Condivisione di volumi tra spazi dei nomi: Dai il benvenuto all'accesso a volumi tra spazi dei nomi](#)".
- Guarda la demo su "[NetAppTV](#)".

Replica dei volumi con SnapMirror

Con Astra Control Provisioner, puoi creare relazioni di mirroring tra un volume di origine su un cluster e il volume di destinazione sul cluster in peering per replicare i dati per il disaster recovery. È possibile utilizzare una definizione di risorsa personalizzata (CRD) con nome per eseguire le seguenti operazioni:

- Creare relazioni di mirroring tra volumi (PVC)
- Rimuovere le relazioni di mirroring tra volumi
- Interrompere le relazioni di mirroring
- Promozione del volume secondario in condizioni di disastro (failover)
- Eseguire la transizione senza perdita di dati delle applicazioni da cluster a cluster (durante failover o migrazioni pianificate)

Prerequisiti per la replica

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

Cluster ONTAP

- **Astra Control Provisioner:** Astra Control Provisioner versione 23.10 o successiva deve esistere sia sui cluster Kubernetes di origine che di destinazione che utilizzano ONTAP come backend.
- **Licenze:** Le licenze asincrone di ONTAP SnapMirror che utilizzano il bundle di protezione dati devono essere attivate sia sul cluster ONTAP di origine che su quello di destinazione. Per ulteriori informazioni, fare riferimento ["Panoramica sulle licenze SnapMirror in ONTAP"](#) a.

Peering

- **Cluster e SVM:** I backend dello storage ONTAP devono essere peering. Per ulteriori informazioni, fare riferimento ["Panoramica del peering di cluster e SVM"](#) a.



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

- **Astra Control Provisioner e SVM:** Le SVM remote in cui è stato eseguito il peering devono essere disponibili per Astra Control Provisioner nel cluster di destinazione.

Driver supportati

- La replica di un volume è supportata per i driver ontap-nas e ontap-san.

Creare un PVC specchiato

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

Fasi

1. Eseguire i seguenti passaggi sul cluster Kubernetes primario:
 - a. Creare 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 StorageClass creato in precedenza.

Esempio

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

- c. Creare una CR MirrorRelationship con informazioni locali.

Esempio

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

Astra Control Provivisioner recupera le informazioni interne del volume e dello stato attuale di data Protection (DP) del volume, quindi popola il campo di stato di MirrorRelationship.

- d. Procurarsi il TridentMirrorRelationship CR per ottenere il nome interno e la SVM del PVC.

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

```

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

```

2. Eseguire i seguenti passaggi sul cluster Kubernetes secondario:

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

Esempio

```

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

```

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

Esempio

```

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

```

Astra Control Provisioner creerà una relazione SnapMirror con il nome della policy di relazione configurata (o default per ONTAP) e la inizializzerà.

- c. Crea un PVC con StorageClass creato in precedenza per agire come secondario (destinazione SnapMirror).

Esempio

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

Astra Control Provisioner controllerà la CRD TridentMirrorRelationship e non creerà il volume se la relazione non esiste. Se esiste una relazione, Astra Control Provisioner garantirà che il nuovo volume FlexVol venga inserito in una SVM a cui viene inviata la SVM remota definita in MirrorRelationship.

Stati di replica dei volumi

Una relazione mirror Trident (TMR) è un CRD che rappresenta un'estremità di una relazione di replica tra PVC. Il TMR di destinazione ha uno stato, che indica ad Astra Control Provisioner lo stato desiderato. Il TMR di destinazione ha i seguenti stati:

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

Promozione del PVC secondario durante un failover non pianificato

Eseguire il seguente passaggio sul cluster Kubernetes secondario:

- Aggiornare il campo `spec.state` di TridentMirrorRelationship a `promoted`.

Promozione del PVC secondario durante un failover pianificato

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

Fasi

1. Sul cluster Kubernetes primario, creare una snapshot del PVC e attendere la creazione dello snapshot.
2. Sul cluster Kubernetes primario, creare SnapshotInfo CR per ottenere dettagli interni.

Esempio

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

3. Nel cluster Kubernetes secondario, aggiornare il campo `spec.state` del `TridentMirrorRelationship` CR a `Promoted` e `spec.promotedSnapshotHandle` come nome interno dello snapshot.
4. Sul cluster Kubernetes secondario, confermare lo stato (campo `status.state`) di `TridentMirrorRelationship` a promosso.

Ripristinare una relazione di mirroring dopo un failover

Prima di ripristinare una relazione di specchiatura, scegliere il lato che si desidera creare come nuovo primario.

Fasi

1. Nel cluster Kubernetes secondario, verificare che i valori per il campo `spec.remoteVolumeHandle` in `TridentMirrorRelationship` siano aggiornati.
2. Sul cluster Kubernetes secondario, aggiornare il campo `spec.mirror` di `TridentMirrorRelationship` a `reestablished`.

Operazioni supplementari

Astra Control Provisioner supporta le seguenti operazioni sui volumi primario e secondario:

Replicare il PVC primario in un nuovo PVC secondario

Assicurarsi di disporre già di un PVC primario e di un PVC secondario.

Fasi

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

Ridimensionare un PVC specchiato, primario o secondario

Il PVC può essere ridimensionato normalmente, ONTAP espanderà automaticamente qualsiasi flexxols di destinazione se la quantità di dati supera le dimensioni correnti.

Rimuovere la replica da un PVC

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

- Eliminare MirrorRelationship sul PVC secondario. Questo interrompe la relazione di replica.
- In alternativa, aggiornare il campo spec.state a *Promoted*.

Eliminazione di un PVC (precedentemente specchiato)

Astra Control Provisioner verifica la presenza di PVC replicati e rilascia la relazione di replica prima di tentare di eliminare il volume.

Eliminare una TMR

L'eliminazione di una TMR su un lato di una relazione specchiata fa sì che la TMR rimanente passi allo stato *promosso* prima che Astra Control Provisioner completi l'eliminazione. Se il TMR selezionato per l'eliminazione è già nello stato *promosso*, non esiste alcuna relazione di mirror esistente e il TMR verrà rimosso e Astra Control Provisioner promuoverà il PVC locale in *ReadWrite*. Questa eliminazione rilascia i metadati SnapMirror per il volume locale in ONTAP. Se in futuro questo volume viene utilizzato in una relazione di mirroring, deve utilizzare un nuovo TMR con uno stato di replica del volume *stabilito* quando si crea la nuova relazione di mirroring.

Aggiorna relazioni mirror quando ONTAP è online

Le relazioni speculari possono essere aggiornate in qualsiasi momento dopo che sono state stabiliti. È possibile utilizzare i state: *promoted* campi o state: *reestablished* per aggiornare le relazioni. Quando si trasferisce un volume di destinazione a un volume *ReadWrite* regolare, è possibile utilizzare *PromotedSnapshotHandle* per specificare uno snapshot specifico su cui ripristinare il volume corrente.

Aggiorna relazioni di mirroring quando ONTAP non è in linea

Puoi utilizzare un CRD per eseguire un update di SnapMirror senza che Astra Control disponga di connettività diretta al cluster ONTAP. Fare riferimento al seguente formato di esempio di TridentActionMirrorUpdate:

Esempio

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

status.state Riflette lo stato del CRD TridentActionMirrorUpdate. Può assumere un valore da *riuscito*, *in corso* o *non riuscito*.

Abilita Astra Control Provisioner

Le versioni 23,10 e successive di Trident includono la possibilità di utilizzare Astra Control Provisioner, che consente agli utenti dotati di licenza Astra Control di accedere a funzionalità avanzate di provisioning dello storage. Astra Control Provisioner fornisce questa funzionalità estesa oltre alle funzionalità standard basate su CSI Astra Trident. È possibile utilizzare questa procedura per abilitare e installare Astra Control Provisioner.

L'abbonamento a Astra Control Service include automaticamente la licenza per l'utilizzo di Astra Control Provisioner.

In arrivo gli update di Astra Control, Astra Control Provisioner sostituirà Astra Trident come provisioner di storage e orchestrator e sarà obbligatorio per l'utilizzo di Astra Control. Per questo motivo, si consiglia vivamente agli utenti di Astra Control di attivare Astra Control Provisioner. Astra Trident continuerà a rimanere open source e ad essere rilasciato, mantenuto, supportato e aggiornato con le nuove CSI e altre funzionalità di NetApp.

Come faccio a sapere se devo abilitare Astra Control Provisioner?

Se si aggiunge un cluster ad Astra Control Service che non ha Astra Trident precedentemente installato, il cluster verrà contrassegnato come **Eligible**. Dopo di voi ["Aggiungi il cluster a Astra Control"](#), Astra Control Provisioner verrà abilitato automaticamente.

Se il cluster non è contrassegnato **Eligible**, verrà contrassegnato **Partially eligible** a causa di uno dei seguenti fattori:

- Sta utilizzando una versione meno recente di Astra Trident
- Sta utilizzando un Astra Trident 23,10 che non ha ancora attivato l'opzione di provisioning
- Si tratta di un tipo di cluster che non consente l'abilitazione automatica

Per **Partially eligible** i casi, usa queste istruzioni per abilitare manualmente Astra Control Provisioner per il tuo cluster.

Choose an Azure Kubernetes Service cluster to enable application data management and the Astra Control storage operator.

⚠ Some Kubernetes cluster(s) below have private networking. [Learn more](#)

Cluster	Location	Eligibility
sandbox-ragnarok-aks-02	centraluseuap	Eligible
sandbox-ragnarok-aks-03	centraluseuap	Partially eligible
sandbox-rstephe2-aks-01	centraluseuap	Eligible

Back Next

Prima di attivare Astra Control Provisioner

Se hai già un Astra Trident senza Astra Control Provisioner e vuoi abilitare Astra Control Provisioner, esegui prima quanto segue:

- **Se Astra Trident è installato, conferma che la sua versione si trovi all'interno di una finestra a quattro release:** Puoi eseguire un aggiornamento diretto a Astra Trident 24,02 con Astra Control Provisioner se il tuo Astra Trident si trova all'interno di una finestra a quattro release della versione 24,02. Ad esempio, puoi eseguire l'upgrade direttamente da Astra Trident 23,04 a 24,02.
- **Confirmi che il tuo cluster ha un'architettura di sistema AMD64:** L'immagine Astra Control Provisioner è fornita in entrambe le architetture CPU AMD64 e ARM64, ma solo AMD64 è supportato da Astra Control.

Fasi

1. Accedere al Registro di sistema dell'immagine di controllo Astra di NetApp:
 - Effettua l'accesso all'interfaccia utente di Astra Control Service e regista l'ID account Astra Control.
 - Selezionare l'icona della figura in alto a destra nella pagina.
 - Selezionare **API access**.
 - Annotare l'ID account.
 - Nella stessa pagina, selezionare **generate API token**, copiare la stringa del token API negli Appunti e salvarla nell'editor.
 - Accedere al registro Astra Control utilizzando il metodo preferito:

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

```
crane auth login cr.astra.netapp.io -u <account-id> -p <api-token>
```

2. (Solo registri personalizzati) attenersi alla seguente procedura per spostare l'immagine nel registro personalizzato. Se non si utilizza un registro, seguire i passaggi dell'operatore Trident nella [sezione successiva](#).



Per i seguenti comandi, puoi utilizzare Podman al posto di Docker. Se si utilizza un ambiente Windows, si consiglia di utilizzare PowerShell.

Docker

- a. Estrarre l'immagine di Astra Control provisioner dal Registro di sistema:



L'immagine estratta non supporta più piattaforme e supporta solo la stessa piattaforma dell'host che ha estratto l'immagine, ad esempio Linux AMD64.

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0  
--platform <cluster platform>
```

Esempio:

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0  
--platform linux/amd64
```

- b. Contrassegnare l'immagine:

```
docker tag cr.astra.netapp.io/astra/trident-acp:24.02.0  
<my_custom_registry>/trident-acp:24.02.0
```

- c. Inviare l'immagine al registro personalizzato:

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

Gru

- a. Copiare il manifesto di Astra Control Provisioner nel registro personalizzato:

```
crane copy cr.astra.netapp.io/astra/trident-acp:24.02.0  
<my_custom_registry>/trident-acp:24.02.0
```

3. Determinare se il metodo di installazione originale di Astra Trident ha utilizzato un.

4. Abilita Astra Control Provisioner in Astra Trident utilizzando il metodo di installazione utilizzato originariamente:

Operatore Astra Trident

- a. ["Scaricare il programma di installazione di Astra Trident ed estrarlo".](#)
- b. Completa questi passaggi se non hai ancora installato Astra Trident o se hai rimosso l'operatore dall'implementazione originale di Astra Trident:
 - i. Creare il CRD:

```
kubectl create -f  
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.1  
6.yaml
```

- ii. Creare lo spazio dei nomi tridente (kubectl create namespace trident) o confermare che lo spazio dei nomi tridente esista ancora (kubectl get all -n trident). Se lo spazio dei nomi è stato rimosso, crearlo di nuovo.
- c. Aggiorna Astra Trident alla versione 24.06.0:



Per i cluster che eseguono Kubernetes 1,24 o versione precedente, utilizzare `bundle_pre_1_25.yaml`. Per i cluster che eseguono Kubernetes 1,25 o versione successiva, utilizzare `bundle_post_1_25.yaml`.

```
kubectl -n trident apply -f trident-installer/deploy/<bundle-  
name.yaml>
```

- d. Verificare che Astra Trident sia in esecuzione:

```
kubectl get torc -n trident
```

Risposta:

NAME	AGE
trident	21m

- e. se si dispone di un registro che utilizza segreti, creare un segreto da utilizzare per estrarre l'immagine di Astra Control Provisioner:

```
kubectl create secret docker-registry <secret_name> -n trident  
--docker-server=<my_custom_registry> --docker-username=<username>  
--docker-password=<token>
```

- f. Modificare il TridentOrchestrator CR e apportare le seguenti modifiche:

```
kubectl edit torc trident -n trident
```

- i. Impostare una posizione del Registro di sistema personalizzata per l'immagine Astra Trident o estrarla dal Registro di sistema Astra Control (tridentImage: <my_custom_registry>/trident:24.02.0 o tridentImage: netapp/trident:24.06.0).
- ii. Attiva Astra Control Provigioner (enableACP: true).
- iii. Impostare la posizione del Registro di sistema personalizzata per l'immagine Astra Control Provigioner o estrarla dal Registro di sistema Astra Control (acpImage: <my_custom_registry>/trident-acp:24.02.0 o acpImage: cr.astra.netapp.io/astra/trident-acp:24.02.0).
- iv. Se è stato stabilito **segreti di estrazione delle immagini** precedentemente in questa procedura, è possibile impostarli qui (imagePullSecrets: - <secret_name>). Usare lo stesso nome segreto che hai stabilito nei passaggi precedenti.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  tridentImage: <registry>/trident:24.06.0
  enableACP: true
  acpImage: <registry>/trident-acp:24.06.0
  imagePullSecrets:
    - <secret_name>
```

- g. Salvare e uscire dal file. Il processo di distribuzione si avvia automaticamente.
- h. Verificare che l'operatore, la distribuzione e i replicaset siano stati creati.

```
kubectl get all -n trident
```



In un cluster Kubernetes dovrebbe esserci solo **un'istanza** dell'operatore. Non creare implementazioni multiple dell'operatore Astra Trident.

- i. Verificare che il trident-acp contenitore sia in funzione e che acpVersion lo stato sia 24.02.0 Installed:

```
kubectl get torc -o yaml
```

Risposta:

```
status:  
  acpVersion: 24.02.0  
  currentInstallationParams:  
    ...  
    acpImage: <registry>/trident-acp:24.02.0  
    enableACP: "true"  
    ...  
    ...  
  status: Installed
```

tridentctl

- ["Scaricare il programma di installazione di Astra Trident ed estrarlo".](#)
- ["Se disponi già di un Astra Trident, disinstallarlo dal cluster che lo ospita".](#)
- Installare Astra Trident con Astra Control Provisioner abilitato (--enable-acp=true):

```
./tridentctl -n trident install --enable-acp=true --acp  
-image=mycustomregistry/trident-acp:24.02
```

- Confermare che Astra Control Provisioner è stato abilitato:

```
./tridentctl -n trident version
```

Risposta:

VERSION	CLIENT VERSION	ACP VERSION	SERVER
24.02.0	24.02.0	24.02.0	24.02.0

Timone

- Se Astra Trident 23.07.1 o versione precedente è installato, ["disinstallazione"](#) l'operatore e altri componenti.
- Se il cluster Kubernetes esegue la versione 1.24 o precedente, elimina psp:

```
kubectl delete psp tridentoperatorpod
```

- Aggiungere il repository Astra Trident Helm:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

d. Aggiornare il grafico Helm:

```
helm repo update netapp-trident
```

Risposta:

```
Hang tight while we grab the latest from your chart
repositories...
...Successfully got an update from the "netapp-trident" chart
repository
Update Complete. □Happy Helming!□
```

e. Elenicare le immagini:

```
./tridentctl images -n trident
```

Risposta:

```
| v1.28.0 | netapp/trident:24.06.0|
|          | docker.io/netapp/trident-
autosupport:24.06|
|          | registry.k8s.io/sig-storage/csi-
provisioner:v4.0.0|
|          | registry.k8s.io/sig-storage/csi-
attacher:v4.5.0|
|          | registry.k8s.io/sig-storage/csi-
resizer:v1.9.3|
|          | registry.k8s.io/sig-storage/csi-
snapshotter:v6.3.3|
|          | registry.k8s.io/sig-storage/csi-node-
driver-registrar:v2.10.0 |
|          | netapp/trident-operator:24.06.0 (optional)
```

f. Assicurarsi che l'operatore di tridente 24.06.0 sia disponibile:

```
helm search repo netapp-trident/trident-operator --versions
```

Risposta:

NAME	CHART VERSION	APP VERSION	
DESCRIPTION			
netapp-trident/trident-operator	100.2406.0	24.06.0	A

g. Utilizzare `helm install` ed eseguire una delle seguenti opzioni che includono queste impostazioni:

- Un nome per la posizione di distribuzione
- La versione di Astra Trident
- Il nome dell'immagine di Astra Control provisioner
- Il flag per abilitare il provisioner
- (Facoltativo) percorso del Registro di sistema locale. Se si utilizza un registro locale, lo "[Immagini Trident](#)" può trovarsi in un registro o in registri diversi, ma tutte le immagini CSI devono trovarsi nello stesso registro.
- Il namespace Trident

Opzioni

- Immagini senza registro

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=cr.astra.netapp.io/astra/trident-
acp:24.06.0 --set enableACP=true --set operatorImage=netapp/trident-
operator:24.06.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.06
--set tridentImage=netapp/trident:24.06.0 --namespace trident
```

- Immagini in uno o più registri

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=<your-registry>:<acp image> --set
enableACP=true --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=netapp/trident-operator:24.06.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.06
--set tridentImage=netapp/trident:24.06.0 --namespace trident
```

È possibile utilizzare `helm list` per esaminare i dettagli dell'installazione come nome, spazio dei nomi, grafico, stato, versione dell'app, e numero di revisione.

Se hai problemi nell'implementazione di Trident utilizzando Helm, esegui questo comando per disinstallare completamente Astra Trident:

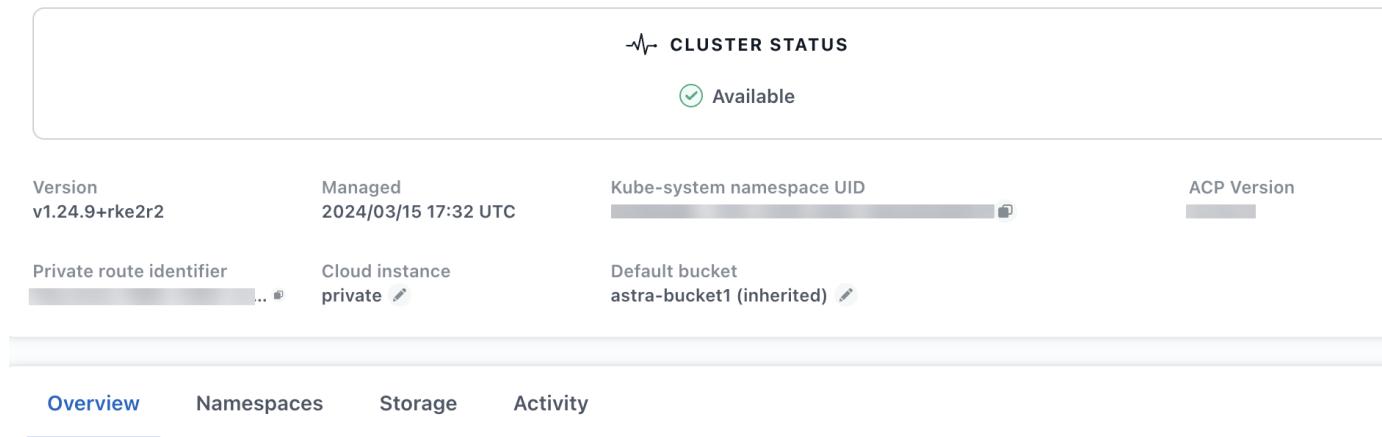
```
./tridentctl uninstall -n trident
```

Non "Rimuovere completamente i CRD Astra Trident" come parte della disinstallazione prima di tentare di abilitare nuovamente Astra Control Provisioner.

Risultato

La funzionalità Astra Control Provisioner è abilitata ed è possibile utilizzare qualsiasi funzionalità disponibile per la versione in esecuzione.

Dopo l'installazione di Astra Control provisioner, il cluster che ospita il provisioner nell'interfaccia utente di Astra Control mostrerà un ACP version numero di versione installata piuttosto che Trident version sul campo.



Per ulteriori informazioni

- ["Documentazione sugli aggiornamenti di Astra Trident"](#)

Utilizzare la topologia CSI

Astra Trident può creare e collegare in modo selettivo i volumi ai nodi presenti in un cluster Kubernetes utilizzando ["Funzionalità topologia CSI"](#) .

Panoramica

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



Ulteriori informazioni sulla funzione topologia CSI "qui" .

Kubernetes offre due esclusive modalità di binding del volume:

- Con VolumeBindingMode impostato su Immediate, Astra Trident crea il volume senza alcuna consapevolezza della topologia. Il binding dei volumi e il provisioning dinamico vengono gestiti quando viene creato il PVC. Questa è l'impostazione predefinita VolumeBindingMode ed è adatta per i cluster che non applicano vincoli di topologia. I volumi persistenti vengono creati senza alcuna dipendenza dai requisiti di pianificazione del pod richiedente.

- Con `VolumeBindingMode` impostato su `WaitForFirstConsumer`, la creazione e l'associazione di un volume persistente per un PVC viene ritardata fino a quando non viene pianificato e creato un pod che utilizza il PVC. In questo modo, i volumi vengono creati per soddisfare i vincoli di pianificazione imposti dai requisiti di topologia.



La `WaitForFirstConsumer` modalità di associazione non richiede etichette topologiche. Questo può essere utilizzato indipendentemente dalla funzionalità topologia CSI.

Di cosa hai bisogno

Per utilizzare la topologia CSI, è necessario disporre di quanto segue:

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

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedead99df",
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:"1e11e4a2108024935ecfcb2912226cedead99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- I nodi nel cluster devono avere etichette che introducano la conoscenza della topologia (`topology.kubernetes.io/region` e `topology.kubernetes.io/zone`). Queste etichette **devono essere presenti sui nodi del cluster** prima dell'installazione di Astra Trident affinché Astra Trident sia consapevole della topologia.

```

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

```

Fase 1: Creazione di un backend compatibile con la topologia

I backend di storage Astra Trident possono essere progettati per eseguire il provisioning selettivo dei volumi in base alle zone di disponibilità. Ogni backend può portare un blocco opzionale `supportedTopologies` che rappresenta un elenco di zone e regioni supportate. Per StorageClasses che utilizzano tale backend, un volume viene creato solo se richiesto da un'applicazione pianificata in una regione/zona supportata.

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

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

```

---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-central1
  topology.kubernetes.io/zone: us-central1-a
- topology.kubernetes.io/region: us-central1
  topology.kubernetes.io/zone: us-central1-b
storage:
- labels:
    workload: production
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
- labels:
    workload: dev
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b

```

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

Fase 2: Definire StorageClasses che siano compatibili con la topologia

In base alle etichette della topologia fornite ai nodi del cluster, è possibile definire StorageClasses in modo da contenere informazioni sulla topologia. In questo modo verranno determinati i pool di storage che fungono da candidati per le richieste PVC effettuate e il sottoinsieme di nodi che possono utilizzare i volumi forniti da Trident.

Vedere il seguente esempio:

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  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 StorageClass fornita sopra, volumeBindingMode è impostato su WaitForFirstConsumer. I PVC richiesti con questa classe di storage non verranno utilizzati fino a quando non saranno referenziati in un pod. E, allowedTopologies fornisce le zone e la regione da utilizzare. netapp-san-us-east1`StorageClass creerà PVC sul `san-backend-us-east1 backend definito sopra.

Fase 3: Creare e utilizzare un PVC

Con StorageClass creato e mappato a un backend, è ora possibile creare PVC.

Fare riferimento all'esempio spec riportato di seguito:

```

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

```

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

```

kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME      STATUS      VOLUME      CAPACITY      ACCESS MODES      STORAGECLASS
AGE
pvc-san   Pending          2s          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

```

Affinché Trident crei un volume e lo leghi al PVC, utilizza 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: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false

```

Questo podSpec richiede a Kubernetes di pianificare il pod sui nodi presenti nella us-east1 regione e di scegliere da qualsiasi nodo presente nelle us-east1-a zone o. us-east1-b

Vedere il seguente output:

```

kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP           NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>          <none>
kubectl get pvc -o wide
NAME        STATUS    VOLUME
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 da includere `supportedTopologies`

I backend preesistenti possono essere aggiornati per includere un elenco di `supportedTopologies` utilizzo di `tridentctl backend update`. Ciò non influisce sui volumi già sottoposti a provisioning e verrà utilizzato solo per i PVC successivi.

Trova ulteriori informazioni

- ["Gestire le risorse per i container"](#)
- ["NodeSelector"](#)
- ["Affinità e anti-affinità"](#)
- ["Contamini e pedaggi"](#)

Lavorare con le istantanee

Le snapshot del volume di Kubernetes dei volumi persistenti (PVS) consentono copie point-in-time dei volumi. Puoi creare una snapshot di un volume creato utilizzando Astra Trident, importare uno snapshot creato all'esterno di Astra Trident, creare un nuovo volume da una snapshot esistente e recuperare i dati del volume da snapshot.

Panoramica

Lo snapshot del volume è supportato da `ontap-nas`, `ontap-nas-flexgroup` `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, e `azure-netapp-files` driver.

Prima di iniziare

Per utilizzare gli snapshot, è necessario disporre di un controller snapshot esterno e di CRD (Custom Resource Definitions). Questa è la responsabilità del Kubernetes orchestrator (ad esempio: Kubeadm, GKE, OpenShift).

Se la distribuzione Kubernetes non include il controller snapshot e i CRD, fare riferimento alla [Implementare un controller per lo snapshot dei volumi](#).



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

Creare un'istantanea del volume

Fasi

1. Creare un `VolumeSnapshotClass`. per ulteriori informazioni, fare riferimento a ["VolumeSnapshotClass"](#)
 - Il driver punti al driver Astra Trident CSI.
 - `deletionPolicy` può essere `Delete` o `Retain`. Quando è impostato su `Retain`, lo snapshot fisico sottostante sul cluster di archiviazione viene conservato anche quando l' `VolumeSnapshot` oggetto viene eliminato.

Esempio

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

2. Creare un'istantanea di un PVC esistente.

Esempi

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

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

- Nell'esempio riportato di seguito viene creato un oggetto `snapshot volume` per un PVC denominato `pvc1` e il nome dello snapshot viene impostato su `pvc1-snap`. Un `VolumeSnapshot` è analogo a un PVC ed è associato a un `VolumeSnapshotContent` oggetto che rappresenta lo snapshot effettivo.

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

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

- È possibile identificare l' `VolumeSnapshotContent` oggetto per `pvc1-snap VolumeSnapshot` descrivendolo. Snapshot Content Name` Identifica l'oggetto `VolumeSnapshotContent` che serve questo snapshot. Il `Ready To Use` parametro indica che l'istantanea può essere utilizzata per creare un nuovo PVC.

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

Creare un PVC da uno snapshot di volume

È possibile utilizzare `dataSource` per creare un PVC utilizzando un `VolumeSnapshot` denominato `<pvc-name>` come origine dei dati. Una volta creato, il PVC può essere collegato a un pod e utilizzato come qualsiasi altro PVC.

 Il PVC verrà creato nello stesso backend del volume di origine. Fare riferimento alla ["KB: La creazione di un PVC da uno snapshot PVC Trident non può essere creata in un backend alternativo"](#).

Nell'esempio seguente viene creato il PVC utilizzando `pvc1-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: pvcl-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io

```

Importare uno snapshot di volume

Astra Trident supporta l' "[Processo Snapshot con pre-provisioning di Kubernetes](#)" per consentire all'amministratore del cluster di creare un `VolumeSnapshotContent` oggetto e importare gli snapshot creati all'esterno di Astra Trident.

Prima di iniziare

Astra Trident deve aver creato o importato il volume principale dello snapshot.

Fasi

1. **Cluster admin:** creare un `VolumeSnapshotContent` oggetto che fa riferimento allo snapshot backend. In questo modo viene avviato il flusso di lavoro delle snapshot in Astra Trident.
 - Specificare il nome dell'istantanea backend in annotations come `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
 - Specifica `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` in `snapshotHandle`. questa è l'unica informazione fornita ad Astra Trident dallo snap-over esterno nella `ListSnapshots` chiamata.



`<volumeSnapshotContentName>` Non può sempre corrispondere al nome dell'istantanea backend a causa di vincoli di denominazione CR.

Esempio

Nell'esempio seguente viene creato un `VolumeSnapshotContent` oggetto che fa riferimento allo snapshot 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>

```

2. **Cluster admin:** creare la VolumeSnapshot CR che fa riferimento all'VolumeSnapshotContent`oggetto. In questo modo viene richiesto l'accesso per utilizzare `VolumeSnapshot in un determinato spazio dei nomi.

Esempio

Nell'esempio seguente viene creata una VolumeSnapshot CR denominata import-snap che fa riferimento alla VolumeSnapshotContent import-snap-content .

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
  # provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content

```

3. **Elaborazione interna (nessuna azione richiesta):** lo snapshot esterno riconosce il nuovo creato ed esegue ListSnapshots la VolumeSnapshotContent chiamata. Astra Trident crea la TridentSnapshot.

- Lo snapshot esterno imposta VolumeSnapshotContent su readyToUse e VolumeSnapshot su true.
- Trident ritorna readyToUse=true.

4. **Qualsiasi utente:** creare un PersistentVolumeClaim per fare riferimento al nuovo VolumeSnapshot, dove il spec.dataSource nome (o spec.dataSourceRef) è il VolumeSnapshot nome.

Esempio

Nell'esempio riportato di seguito viene creato un PVC che fa riferimento alla VolumeSnapshot 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
```

Ripristinare i dati del volume utilizzando le snapshot

La directory dello snapshot è nascosta per impostazione predefinita in modo da facilitare la massima compatibilità dei volumi sottoposti a provisioning mediante i `ontap-nas` driver e. `ontap-nas-economy`. Abilitare la `.snapshot` directory per il ripristino diretto dei dati dagli snapshot.

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

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



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

La directory dello snapshot è nascosta per impostazione predefinita in modo da facilitare la massima compatibilità dei volumi sottoposti a provisioning mediante i `ontap-nas` driver e. `ontap-nas-economy`. Abilitare la `.snapshot` directory per il ripristino diretto dei dati dagli snapshot.



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

Ripristino del volume in-place da uno snapshot

Astra Control Provisioner consente il ripristino rapido e in-place dei volumi da uno snapshot utilizzando il `TridentActionSnapshotRestore` CR (TASR). Questo CR funziona come un'azione imperativa di

Kubernetes e non persiste al termine dell'operazione.

Astra Control Provisioner supporta il ripristino delle istantanee su `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `gcp-cvs`, e `solidfire-san` driver.

Prima di iniziare

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

- Verificare che lo stato del PVC sia limitato.

```
kubectl get pvc
```

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

```
kubectl get vs
```

Fasi

1. Creare TASR CR. In questo esempio viene creata una CR per PVC `pvc1` e snapshot volume `pvc1-snapshot`.

```
cat tasr-pvc1-snapshot.yaml

apiVersion: v1
kind: TridentActionSnapshotRestore
metadata:
  name: this-doesnt-matter
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. Applicare la CR per eseguire il ripristino dall'istantanea. Nell'esempio riportato di seguito vengono ripristinati gli snapshot `pvc1`.

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

tridentactionsnapshotrestore.trident.netapp.io/this-doesnt-matter
created
```

Risultati

Astra Control Provisioner ripristina i dati dalla snapshot. È possibile verificare lo stato di ripristino dello snapshot.

```

kubectl get tasr -o yaml

apiVersion: v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: this-doesnt-matter
    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, Astra Control provisioner non ritenta automaticamente l'operazione in caso di guasto. Sarà necessario eseguire nuovamente l'operazione.
- Gli utenti Kubernetes senza accesso amministrativo potrebbero dover essere autorizzati dall'amministratore a creare una TASR CR nel namespace delle applicazioni.

Eliminare un PV con gli snapshot associati

Quando si elimina un volume persistente con snapshot associate, il volume Trident corrispondente viene aggiornato a uno stato di eliminazione. Rimuovere le snapshot del volume per eliminare il volume Astra Trident.

Implementare un controller per lo snapshot dei volumi

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

Fasi

1. Creare CRD snapshot di 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. Creare 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, aprire `deploy/kubernetes/snapshot-controller/rbac-
snapshot-controller.yaml` e aggiornare lo namespace spazio dei nomi.

Link correlati

- ["Snapshot dei volumi"](#)
- ["VolumeSnapshotClass"](#)

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 COMMERCIALITÀ 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.