



# **Provisionierung und Management von Volumes**

**Trident**

NetApp

February 02, 2026

# Inhalt

Provisionierung und Management von Volumes .....	1
Bereitstellen eines Volumes .....	1
Überblick .....	1
Erstellen Sie die PVC .....	1
Erweitern Sie Volumes .....	5
Erweitern Sie ein iSCSI-Volume .....	5
Erweitern Sie ein FC-Volume .....	9
Erweitern Sie ein NFS-Volume .....	13
Volumes importieren .....	16
Überblick und Überlegungen .....	16
Importieren Sie ein Volume .....	17
Beispiele .....	19
Passen Sie Volume-Namen und -Beschriftungen an .....	26
Bevor Sie beginnen .....	26
Einschränkungen .....	26
Wichtige Verhaltensweisen anpassbarer Volumennamen .....	27
Beispiele für die Backend-Konfiguration mit Namensvorlage und Beschriftungen .....	27
Beispiele für Namensvorlagen .....	28
Zu berücksichtigende Aspekte .....	29
Ein NFS-Volume kann über Namespaces hinweg genutzt werden .....	29
Funktionen .....	29
Schnellstart .....	30
Konfigurieren Sie die Namensräume für Quelle und Ziel .....	31
Löschen eines freigegebenen Volumes .....	32
Nutzung <code>tridentctl get</code> Zum Abfragen von untergeordneten Volumes .....	32
Einschränkungen .....	33
Finden Sie weitere Informationen .....	33
Volumes können in Namespaces geklont werden .....	33
Voraussetzungen .....	33
Schnellstart .....	33
Konfigurieren Sie die Namensräume für Quelle und Ziel .....	34
Einschränkungen .....	36
Replizieren Sie Volumes mit SnapMirror .....	36
Replikationsvoraussetzungen .....	36
Erstellen Sie eine gespiegelte PVC .....	37
Volume-Replikationsstatus .....	40
Fördern Sie die sekundäre PVC während eines ungeplanten Failover .....	40
Fördern Sie die sekundäre PVC während eines geplanten Failover .....	40
Stellen Sie nach einem Failover eine gespiegelte Beziehung wieder her .....	41
Zusätzliche Vorgänge .....	41
Aktualisieren Sie Spiegelbeziehungen, wenn ONTAP online ist .....	42
Aktualisieren Sie Spiegelbeziehungen, wenn ONTAP offline ist .....	42
Verwenden Sie die CSI-Topologie .....	42

Überblick .....	42
Schritt 1: Erstellen Sie ein Topologieorientiertes Backend.....	44
Schritt: Definition von StorageClasses, die sich der Topologie bewusst sind .....	46
Schritt 3: Erstellen und verwenden Sie ein PVC .....	47
Aktualisieren Sie Back-Ends, um einzuschließen supportedTopologies .....	50
Weitere Informationen .....	50
Arbeiten Sie mit Snapshots .....	50
Überblick .....	50
Erstellen eines Volume-Snapshots .....	51
Erstellen Sie eine PVC aus einem Volume-Snapshot .....	52
Importieren Sie einen Volume-Snapshot .....	53
Stellen Sie Volume-Daten mithilfe von Snapshots wieder her .....	55
In-Place-Volume-Wiederherstellung aus einem Snapshot.....	55
Löschen Sie ein PV mit den zugehörigen Snapshots .....	57
Stellen Sie einen Volume-Snapshot-Controller bereit .....	57
Weiterführende Links .....	58
Arbeiten mit Volumegruppen-Snapshots .....	58
Erstellen von Volumegruppen-Snapshots .....	59
Wiederherstellen von Volumedaten mithilfe eines Gruppen-Snapshots .....	60
In-Place-Volume-Wiederherstellung aus einem Snapshot.....	61
Löschen eines PV mit zugehörigen Gruppen-Snapshots .....	61
Stellen Sie einen Volume-Snapshot-Controller bereit .....	61
Weiterführende Links .....	62

# Provisionierung und Management von Volumes

## Bereitstellen eines Volumes

Erstellen Sie ein PersistentVolumeClaim (PVC), das die konfigurierte Kubernetes StorageClass verwendet, um Zugriff auf das PV anzufordern. Anschließend können Sie das PV an einem Pod montieren.

### Überblick

A "[PersistentVolumeClaim](#)" (PVC) ist eine Anforderung für den Zugriff auf das PersistentVolume auf dem Cluster.

Die PVC kann so konfiguriert werden, dass eine Speicherung einer bestimmten Größe oder eines bestimmten Zugriffsmodus angefordert wird. Mithilfe der zugehörigen StorageClass kann der Clusteradministrator mehr als die Größe des PersistentVolume und den Zugriffsmodus steuern, z. B. die Performance oder das Service-Level.

Nachdem Sie die PVC erstellt haben, können Sie das Volume in einem Pod einbinden.

### Erstellen Sie die PVC

#### Schritte

1. Erstellen Sie die PVC.

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

2. Überprüfen Sie den PVC-Status.

```
kubectl get pvc
```

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

1. Mounten Sie das Volume in einem Pod.

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



Sie können den Fortschritt mit überwachen `kubectl get pod --watch`.

2. Vergewissern Sie sich, dass das Volume auf gemountet ist `/my/mount/path`.

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

3. Sie können den Pod jetzt löschen. Die Pod Applikation wird nicht mehr existieren, aber das Volume bleibt erhalten.

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

## Beispielmanifeste

## PersistentVolumeClaim-Beispielmanifeste

Diese Beispiele zeigen grundlegende PVC-Konfigurationsoptionen.

### PVC mit RWO-Zugang

Dieses Beispiel zeigt eine grundlegende PVC mit RWO-Zugriff, die einer StorageClass mit dem Namen zugeordnet ist basic-csi.

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

### PVC mit NVMe/TCP

Dieses Beispiel zeigt eine grundlegende PVC für NVMe/TCP mit RWO-Zugriff, die einer StorageClass mit dem Namen zugeordnet ist protection-gold.

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

## Pod-Manifest-Proben

Diese Beispiele zeigen grundlegende Konfigurationen zum Anschließen der PVC an einen Pod.

### Basiskonfiguration

```
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
```

### Grundlegende NVMe/TCP-Konfiguration

```
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
```

Einzelheiten zur Interaktion von Storage-Klassen mit den PersistentVolumeClaim Parametern und zur Steuerung, wie Trident Volumes provisioniert, finden Sie unter "[Kubernetes und Trident Objekte](#)".

# Erweitern Sie Volumes

Trident bietet Kubernetes-Benutzern die Möglichkeit, ihre Volumes nach der Erstellung zu erweitern. Hier finden Sie Informationen zu den Konfigurationen, die zum Erweitern von iSCSI-, NFS-, SMB-, NVMe/TCP- und FC-Volumes erforderlich sind.

## Erweitern Sie ein iSCSI-Volume

Sie können ein iSCSI Persistent Volume (PV) mithilfe der CSI-provisionierung erweitern.



Die Erweiterung des iSCSI-Volumes wird von unterstützt `ontap-san`, `ontap-san-economy`, `solidfire-san` Treiber und erfordert Kubernetes 1.16 und höher.

### Schritt: Storage Class für Volume-Erweiterung konfigurieren

Bearbeiten Sie die StorageClass-Definition, um die festzulegen `allowVolumeExpansion` Feld an `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
```

Bearbeiten Sie für eine bereits vorhandene StorageClass, um die einzuschließen `allowVolumeExpansion` Parameter.

### Schritt 2: Erstellen Sie ein PVC mit der von Ihnen erstellten StorageClass

Bearbeiten Sie die PVC-Definition, und aktualisieren Sie die `spec.resources.requests.storage` Um die neu gewünschte Größe zu reflektieren, die größer als die ursprüngliche Größe sein muss.

```
cat pvc-ontapsan.yaml
```

```

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

```

Trident erstellt ein persistentes Volume (PV) und verknüpft es mit diesem 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

```

### Schritt 3: Definieren Sie einen Behälter, der das PVC befestigt

Schließen Sie das PV an einen Pod an, um die Größe zu ändern. Beim Ändern der Größe eines iSCSI-PV gibt es zwei Szenarien:

- Wenn das PV mit einem Pod verbunden ist, erweitert Trident das Volume im Storage-Back-End, scannt das Gerät erneut und skaliert das Dateisystem.
- Beim Versuch, die Größe eines nicht verbundenen PV zu ändern, erweitert Trident das Volume auf dem Speicher-Back-End. Nachdem die PVC an einen Pod gebunden ist, lässt Trident das Gerät neu in die Größe des Dateisystems einarbeiten. Kubernetes aktualisiert dann die PVC-Größe, nachdem der Expand-Vorgang erfolgreich abgeschlossen ist.

In diesem Beispiel wird ein POD erstellt, der die verwendet `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
```

#### Schritt 4: Erweitern Sie das PV

Um die Größe des PV zu ändern, das von 1Gi auf 2Gi erstellt wurde, bearbeiten Sie die PVC-Definition und aktualisieren Sie die `spec.resources.requests.storage` Bis 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
# ...

```

## Schritt 5: Validieren Sie die Erweiterung

Sie können die korrekt bearbeitete Erweiterung validieren, indem Sie die Größe der PVC, des PV und des Trident Volume überprüfen:

```

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
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      |
+-----+-----+-----+
+-----+-----+-----+

```

## Erweitern Sie ein FC-Volume

Sie können ein FC Persistent Volume (PV) mit der CSI-provisionierung erweitern.



FC-Volume-Erweiterung wird vom Treiber unterstützt `ontap-san` und erfordert Kubernetes 1.16 und höher.

### Schritt: Storage Class für Volume-Erweiterung konfigurieren

Bearbeiten Sie die StorageClass-Definition, um die festzulegen `allowVolumeExpansion` Feld an `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

```

Bearbeiten Sie für eine bereits vorhandene StorageClass, um die einzuschließen allowVolumeExpansion Parameter.

### Schritt 2: Erstellen Sie ein PVC mit der von Ihnen erstellten StorageClass

Bearbeiten Sie die PVC-Definition, und aktualisieren Sie die spec.resources.requests.storage Um die neu gewünschte Größe zu reflektieren, die größer als die ursprüngliche Größe sein muss.

```
cat pvc-ontapsan.yaml
```

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

Trident erstellt ein persistentes Volume (PV) und verknüpft es mit diesem 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
```

### Schritt 3: Definieren Sie einen Behälter, der das PVC befestigt

Schließen Sie das PV an einen Pod an, um die Größe zu ändern. Beim Ändern der Größe eines FC-PV gibt es zwei Szenarien:

- Wenn das PV mit einem Pod verbunden ist, erweitert Trident das Volume im Storage-Back-End, scannt das Gerät erneut und skaliert das Dateisystem.
- Beim Versuch, die Größe eines nicht verbundenen PV zu ändern, erweitert Trident das Volume auf dem Speicher-Back-End. Nachdem die PVC an einen Pod gebunden ist, lässt Trident das Gerät neu in die

Größe des Dateisystems einarbeiten. Kubernetes aktualisiert dann die PVC-Größe, nachdem der Expand-Vorgang erfolgreich abgeschlossen ist.

In diesem Beispiel wird ein POD erstellt, der die verwendet `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
```

#### Schritt 4: Erweitern Sie das PV

Um die Größe des PV zu ändern, das von 1Gi auf 2Gi erstellt wurde, bearbeiten Sie die PVC-Definition und aktualisieren Sie die `spec.resources.requests.storage` Bis 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
# ...

```

## Schritt 5: Validieren Sie die Erweiterung

Sie können die korrekt bearbeitete Erweiterung validieren, indem Sie die Größe der PVC, des PV und des Trident Volume überprüfen:

```

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      |
+-----+-----+-----+
+-----+-----+-----+

```

## Erweitern Sie ein NFS-Volume

Trident unterstützt die Volumenerweiterung für NFS-PVs, die auf `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, Und `azure-netapp-files` Backends.

### Schritt: Storage Class für Volume-Erweiterung konfigurieren

Um die Größe eines NFS PV zu ändern, muss der Administrator zunächst die Storage-Klasse konfigurieren, um die Volume-Erweiterung durch Einstellen der zu ermöglichen `allowVolumeExpansion` Feld an `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

```

Wenn Sie bereits eine Storage-Klasse ohne diese Option erstellt haben, können Sie die vorhandene Storage-

Klasse einfach mit bearbeiten `kubectl edit storageclass` Um eine Volume-Erweiterung zu ermöglichen.

## Schritt 2: Erstellen Sie ein PVC mit der von Ihnen erstellten StorageClass

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

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

Trident sollte für dieses PVC ein 20 MiB NFS PV erstellen:

```
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
```

## Schritt 3: Erweitern Sie das PV

Um die Größe des neu erstellten 20 MiB PV auf 1 GiB zu ändern, bearbeiten Sie den PVC und legen Sie fest `spec.resources.requests.storage` bis 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
# ...

```

#### Schritt 4: Validieren Sie die Erweiterung

Sie können die Größe der korrekt bearbeiteten Größe validieren, indem Sie die Größe der PVC, des PV und des Trident Volume überprüfen:

```

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        |
+-----+-----+-----+
+-----+-----+-----+

```

## Volumes importieren

Sie können vorhandene Speichervolumes als Kubernetes PV mit `tridentctl import` importieren oder indem Sie einen Persistent Volume Claim (PVC) mit Trident-Importannotationen erstellen.

### Überblick und Überlegungen

Sie können ein Volume in Trident importieren, um:

- Containerisierung einer Applikation und Wiederverwendung des vorhandenen Datensatzes
- Verwenden Sie einen Klon eines Datensatzes für eine kurzlebige Applikation
- Wiederherstellung eines fehlerhaften Kubernetes-Clusters
- Migration von Applikationsdaten bei der Disaster Recovery

### Überlegungen

Lesen Sie vor dem Importieren eines Volumes die folgenden Überlegungen durch.

- Trident kann nur ONTAP-Volumes vom Typ RW (Lesen/Schreiben) importieren. Volumes im DP-Typ

(Datensicherung) sind SnapMirror Ziel-Volumes. Sie sollten die Spiegelungsbeziehung unterbrechen, bevor Sie das Volume in Trident importieren.

- Wir empfehlen, Volumes ohne aktive Verbindungen zu importieren. Um ein aktiv verwendetes Volume zu importieren, klonen Sie das Volume, und führen Sie dann den Import durch.



Dies ist besonders für Block-Volumes wichtig, da Kubernetes die vorherige Verbindung nicht mitbekommt und problemlos ein aktives Volume an einen Pod anbinden kann. Dies kann zu Datenbeschädigungen führen.

- Obwohl StorageClass auf einer PVC angegeben werden muss, verwendet Trident diesen Parameter beim Import nicht. Während der Volume-Erstellung werden Storage-Klassen eingesetzt, um basierend auf den Storage-Merkmalen aus verfügbaren Pools auszuwählen. Da das Volume bereits vorhanden ist, ist beim Import keine Poolauswahl erforderlich. Daher schlägt der Import auch dann nicht fehl, wenn das Volume auf einem Back-End oder Pool vorhanden ist, das nicht mit der in der PVC angegebenen Speicherklasse übereinstimmt.
- Die vorhandene Volumegröße wird in der PVC ermittelt und festgelegt. Nachdem das Volumen vom Speichertreiber importiert wurde, wird das PV mit einem ClaimRef an die PVC erzeugt.
  - Die Rückgewinnungsrichtlinie ist zunächst auf festgelegt retain im PV. Nachdem Kubernetes die PVC und das PV erfolgreich bindet, wird die Zurückgewinnungsrichtlinie aktualisiert und an die Zurückgewinnungsrichtlinie der Storage-Klasse angepasst.
  - Wenn die Richtlinie zur Zurückgewinnung der Storage-Klasse lautet delete, Das Speichervolumen wird gelöscht, wenn das PV gelöscht wird.
- Standardmäßig verwaltet Trident die PVC und benennt das FlexVol volume und die LUN im Backend um. Du kannst die --no-manage Flag zum Importieren eines nicht verwalteten Volumes und die --no-rename Flagge, um den Datenträgernamen beizubehalten.
  - --no-manage\* - Wenn Sie die --no-manage Laut dieser Meldung führt Trident während des gesamten Lebenszyklus der Objekte keine zusätzlichen Operationen an der PVC oder PV durch. Das Speichervolume wird beim Löschen des PV nicht gelöscht, und auch andere Operationen wie das Klonen von Volumes und das Ändern der Volume-Größe werden ignoriert.
  - --no-rename\* - Wenn Sie die --no-rename Mit dieser Option behält Trident beim Importieren von Volumes den bestehenden Volume-Namen bei und verwaltet den Lebenszyklus der Volumes. Diese Option wird nur für die ontap-nas , ontap-san (einschließlich ASA r2-Systeme) und ontap-san-economy Fahrer.



Diese Optionen sind nützlich, wenn Sie Kubernetes für containerisierte Workloads verwenden möchten, ansonsten aber den Lebenszyklus des Speichervolumens außerhalb von Kubernetes verwalten möchten.

- Der PVC und dem PV wird eine Anmerkung hinzugefügt, die einem doppelten Zweck dient, anzugeben, dass das Volumen importiert wurde und ob PVC und PV verwaltet werden. Diese Anmerkung darf nicht geändert oder entfernt werden.

## Importieren Sie ein Volume

Sie können ein Volume entweder mit `tridentctl import` oder durch das Erstellen einer PVC mit Trident-Importanmerkungen importieren.



Wenn Sie PVC-Anmerkungen verwenden, müssen Sie `tridentctl` nicht herunterladen oder verwenden, um das Volume zu importieren.

## Verwendung von tridentctl

### Schritte

1. Erstellen Sie eine PVC-Datei (zum Beispiel pvc.yaml), die zur Erstellung des PVC verwendet wird. Die PVC-Datei sollte name, namespace, accessModes und storageClassName enthalten. Optional können Sie unixPermissions in Ihrer PVC-Definition angeben.

Im Folgenden finden Sie ein Beispiel für eine Mindestspezifikation:

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



Geben Sie nur die erforderlichen Parameter an. Zusätzliche Parameter wie PV-Name oder Volume-Größe können dazu führen, dass der Importbefehl fehlschlägt.

2. Verwenden Sie die tridentctl import Befehl zur Angabe des Namens des Trident -Backends, das das Volume enthält, und des Namens, der das Volume auf dem Speicher eindeutig identifiziert (z. B. ONTAP FlexVol, Element Volume). Der -f Ein Argument ist erforderlich, um den Pfad zur PVC-Datei anzugeben.

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

## Verwendung von PVC-Anmerkungen

### Schritte

1. Erstellen Sie eine PVC-YAML-Datei (zum Beispiel pvc.yaml) mit den erforderlichen Trident-Importannotationen. Die PVC-Datei sollte Folgendes enthalten:
  - name and namespace in Metadaten
  - accessModes, resources.requests.storage, und storageClassName in Spezifikation
  - Anmerkungen:
    - trident.netapp.io/importOriginalName: Volumenname auf dem Backend
    - trident.netapp.io/importBackendUUID: Backend-UUID, wo das Volume existiert
    - trident.netapp.io/notManaged (*Optional*): Auf "true" für nicht verwaltete Volumes festlegen. Standardwert ist "false".

Im Folgenden finden Sie eine Beispielspezifikation für den Import eines verwalteten Volumes:

```

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. Wenden Sie die PVC YAML-Datei auf Ihren Kubernetes-Cluster an:

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

Trident wird das Volumen automatisch importieren und es an das PVC binden.

## Beispiele

Lesen Sie die folgenden Beispiele für den Import von Volumes für unterstützte Treiber.

### ONTAP NAS und ONTAP NAS FlexGroup

Trident unterstützt den Import von Volumes mit den `ontap-nas` Treibern und `ontap-nas-flexgroup`.

- Trident unterstützt keinen Volumenimport mit dem `ontap-nas-economy` Treiber.
- Der `ontap-nas` Und `ontap-nas-flexgroup` Treiber erlauben keine doppelten Volume-Namen.

Jedes mit dem Treiber erstellte Volume `ontap-nas` ist eine FlexVol volume im ONTAP Cluster. Der Import von FlexVol-Volumes mit dem `ontap-nas` Treiber funktioniert gleich. FlexVol Volumes, die bereits in einem ONTAP-Cluster vorhanden sind, können als PVC importiert werden `ontap-nas`. Ebenso können FlexGroup-Volumes als PVCs importiert werden `ontap-nas-flexgroup`.

### ONTAP NAS-Beispiele mit `tridentctl`

Die folgenden Beispiele zeigen, wie verwaltete und nicht verwaltete Volumes mit `tridentctl` importiert werden.

## Gemanagtes Volume

Im folgenden Beispiel wird ein Volume mit dem Namen importiert `managed_volume` Auf einem Backend mit dem Namen `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     |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

## Nicht verwaltetes Volume

Bei Verwendung des `--no-manage` Arguments benennt Trident das Volume nicht um.

Das folgende Beispiel importiert `unmanaged_volume` Auf dem `ontap_nas` Back-End:

```
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 NAS-Beispiele mit PVC-Annotationen

Die folgenden Beispiele zeigen, wie verwaltete und nicht verwaltete Volumes mithilfe von PVC-Annotationen importiert werden.

## Gemanagtes Volume

Das folgende Beispiel importiert ein 1Gi ontap-nas Volume namens `ontap_volume1` vom Backend `81abcb27-ea63-49bb-b606-0a5315ac5f21` mit RWO-Zugriffsmodus, der mithilfe von PVC-Annotationen festgelegt ist:

```
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>
```

## Nicht verwaltetes Volume

Das folgende Beispiel importiert ein 1Gi ontap-nas Volume mit dem Namen `ontap-volume2` vom Backend `34abcb27-ea63-49bb-b606-0a5315ac5f34` mit festgelegtem RWO-Zugriffsmodus mithilfe von PVC-Annotationen:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <umanned-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>
```

## ONTAP SAN

Trident unterstützt den Volumenimport mithilfe der `ontap-san` (iSCSI, NVMe/TCP und FC) und `ontap-san-economy` Treiber.

Trident kann ONTAP SAN FlexVol -Volumes importieren, die eine einzelne LUN enthalten. Dies steht im Einklang mit der `ontap-san` Treiber, der für jedes PVC ein FlexVol volume und eine LUN innerhalb des FlexVol volume erstellt. Trident importiert das FlexVol volume und verknüpft es mit der PVC-Definition. Trident kann importieren `ontap-san-economy` Volumes, die mehrere LUNs enthalten.

Die folgenden Beispiele zeigen, wie verwaltete und nicht verwaltete Volumes importiert werden:

## Gemanagtes Volume

Für verwaltete Volumes benennt Trident die FlexVol volume in das Format und die LUN in der FlexVol volume in lun0 um pvc-<uuid>.

Im folgenden Beispiel werden die auf dem Backend vorhandenen FlexVol volume ontap\_san\_default importiert ontap-san-managed:

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

## Nicht verwaltetes Volume

Das folgende Beispiel importiert unmanaged\_example\_volume Auf dem ontap\_san Back-End:

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

Wenn LUNS Initiatorgruppen zugeordnet sind, die einen IQN mit einem Kubernetes-Node-IQN teilen, wie im folgenden Beispiel dargestellt, erhalten Sie die Fehlermeldung: LUN already mapped to initiator(s) in this group. Sie müssen den Initiator entfernen oder die Zuordnung der LUN aufheben, um das Volume zu importieren.

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

## Element

Trident unterstützt NetApp Element-Software und NetApp HCI-Volume-Import mit dem `solidfire-san` Treiber.



Der Elementtreiber unterstützt doppelte Volume-Namen. Trident gibt jedoch einen Fehler zurück, wenn es doppelte Volume-Namen gibt. Um dies zu umgehen, klonen Sie das Volume, geben Sie einen eindeutigen Volume-Namen ein und importieren Sie das geklonte Volume.

Im folgenden Beispiel wird ein importiert `element-managed` Volume am 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 Dateien

Trident unterstützt den Import von Volumes mithilfe des `azure-netapp-files` Treibers.



Um ein Azure NetApp Files-Volume zu importieren, identifizieren Sie das Volume anhand seines Volume-Pfads. Der Volume-Pfad ist der Teil des Exportpfades des Volumes nach dem `:`. Beispiel: Wenn der Mount-Pfad lautet `10.0.0.2:/importvol1`, Der Volume-Pfad ist `importvol1`.

Im folgenden Beispiel wird ein importiert `azure-netapp-files` Volume am Backend `azurenetaffiles_40517` Mit dem Volume-Pfad `importvol1`.

```
tridentctl import volume azurenappfiles_40517 importvol1 -f <path-to-pvc-file> -n trident

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

## Google Cloud NetApp Volumes

Trident unterstützt den Import von Volumes mithilfe des `google-cloud-netapp-volumes` Treibers.

Das folgende Beispiel importiert ein Volume auf Backend `backend-tbc-gcnv1` mit dem Volume `testvoleasiaeast1`.

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

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

Im folgenden Beispiel wird ein Volume importiert `google-cloud-netapp-volumes`, wenn zwei Volumes in derselben Region vorhanden sind:

```

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
|
+-----+-----+
+-----+-----+
+-----+-----+

```

## Passen Sie Volume-Namen und -Beschriftungen an

Mit Trident können Sie Volumes, die Sie erstellen, aussagekräftige Namen und Labels zuweisen. So können Sie Volumes leichter identifizieren und ihren jeweiligen Kubernetes-Ressourcen (PVCs) zuweisen. Sie können auch Vorlagen auf Backend-Ebene definieren, um benutzerdefinierte Volume-Namen und benutzerdefinierte Labels zu erstellen. Alle Volumes, die Sie erstellen, importieren oder klonen, werden an die Vorlagen angepasst.

### Bevor Sie beginnen

Anpassbare Volumennamen und Beschriftungen unterstützen:

- Volume-Erstellung, -Import und -Klonen
- Im Falle des `ontap-nas-economy` Beim Treiber entspricht nur der Name des Qtree-Volumes der Namensvorlage.
- Im Falle des `ontap-san-economy` Beim Treiber entspricht nur der LUN-Name der Namensvorlage.

### Einschränkungen

- Benutzerdefinierte Volumennamen sind nur mit ONTAP -Treibern für lokale Installationen kompatibel.
- Benutzerdefinierte Etiketten werden nur für die `ontap-san`, `ontap-nas`, Und `ontap-nas-flexgroup` Fahrer.
- Benutzerdefinierte Volumennamen gelten nicht für bereits vorhandene Volumen.

## Wichtige Verhaltensweisen anpassbarer Volumennamen

- Wenn ein Fehler aufgrund einer ungültigen Syntax in einer Namensvorlage auftritt, schlägt die Back-End-Erstellung fehl. Wenn jedoch die Vorlagenapplikation fehlschlägt, wird das Volume gemäß der bestehenden Namenskonvention benannt.
- Storage-Präfix ist nicht anwendbar, wenn ein Volume mit einer Namensvorlage aus der Back-End-Konfiguration benannt wird. Jeder gewünschte Präfixwert kann direkt zur Vorlage hinzugefügt werden.

## Beispiele für die Backend-Konfiguration mit Namensvorlage und Beschriftungen

Benutzerdefinierte Namensvorlagen können auf Root- und/oder Poolebene definiert werden.

### Beispiel für die Stammebene

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

## Beispiel auf Poolebene

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

## Beispiele für Namensvorlagen

### Beispiel 1:

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

### Beispiel 2:

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

## Zu berücksichtigende Aspekte

- Bei Volumenimporten werden die Etiketten nur aktualisiert, wenn das vorhandene Volume über Etiketten in einem bestimmten Format verfügt. Zum Beispiel: {"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"} }.
- Im Fall des Imports von verwalteten Volumes folgt der Name des Volumes der Namensvorlage, die in der Backend-Definition auf Root-Ebene definiert wurde.
- Trident unterstützt die Verwendung eines Slice-Operators mit dem Speicherpräfix nicht.
- Wenn die Vorlagen nicht zu eindeutigen Volume-Namen führen, fügt Trident einige zufällige Zeichen an, um eindeutige Volume-Namen zu erstellen.
- Wenn der benutzerdefinierte Name für ein NAS-Economy-Volume 64 Zeichen lang ist, benennt Trident die Volumes entsprechend der bestehenden Namenskonvention. Bei allen anderen ONTAP-Treibern schlägt die Erstellung des Volumes fehl, wenn der Datenträgername das Limit für den Namen überschreitet.

## Ein NFS-Volume kann über Namespaces hinweg genutzt werden

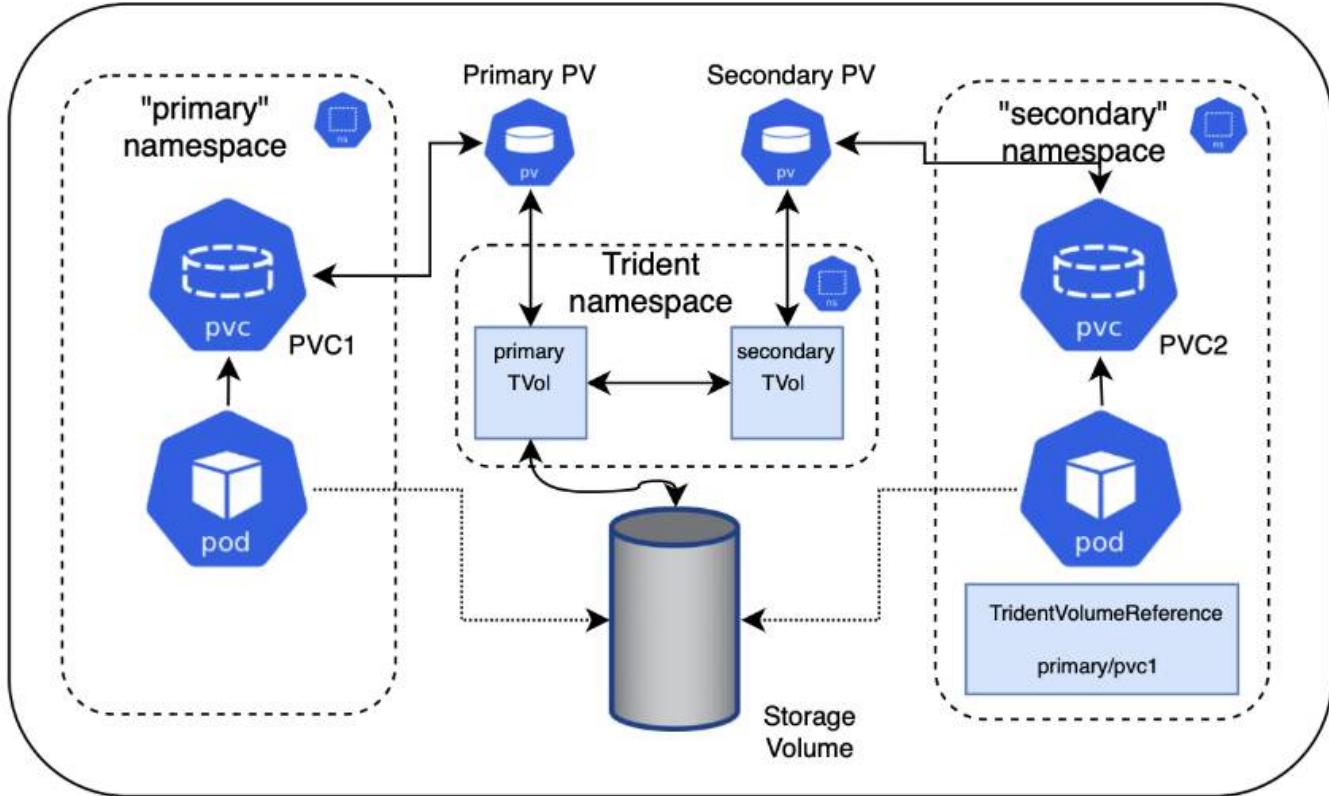
Mit Trident können Sie ein Volume in einem primären Namespace erstellen und es in einem oder mehreren sekundären Namespaces teilen.

### Funktionen

Mit dem TridentVolumeReference CR können Sie ReadWriteMany (RWX) NFS-Volumes sicher über einen oder mehrere Kubernetes-Namespace freigeben. Diese native Kubernetes-Lösung bietet folgende Vorteile:

- Mehrere Stufen der Zugriffssteuerung zur Sicherstellung der Sicherheit
- Funktioniert mit allen Trident NFS-Volume-Treibern
- Tridentctl oder andere nicht-native Kubernetes-Funktionen sind nicht von Bedeutung

Dieses Diagramm zeigt die NFS-Volume-Freigabe über zwei Kubernetes-Namespace.



## Schnellstart

Sie können in nur wenigen Schritten NFS-Volume Sharing einrichten.

**1**

**Konfigurieren Sie die PVC-Quelle für die gemeinsame Nutzung des Volumes**

Der Eigentümer des Quell-Namespace erteilt die Berechtigung, auf die Daten im Quell-PVC zuzugreifen.

**2**

**Berechtigung zum Erstellen eines CR im Ziel-Namespace gewähren**

Der Clusteradministrator erteilt dem Eigentümer des Ziel-Namespace die Berechtigung, das TridentVolumeReference CR zu erstellen.

**3**

**Erstellen Sie im Ziel-Namespace tridentVolumeReference**

Der Eigentümer des Ziel-Namespace erstellt das TridentVolumeReference CR, um sich auf das Quell-PVC zu beziehen.

**4**

**Erstellen Sie das untergeordnete PVC im Ziel-Namespace**

Der Eigentümer des Ziel-Namespace erstellt das untergeordnete PVC, um die Datenquelle aus dem Quell-PVC zu verwenden.

## Konfigurieren Sie die Namensräume für Quelle und Ziel

Um die Sicherheit zu gewährleisten, erfordert die Namespace-übergreifende Freigabe Zusammenarbeit und Aktion durch den Eigentümer des Quell-Namespace, den Cluster-Administrator und den Ziel-Namespace-Eigentümer. In jedem Schritt wird die Benutzerrolle festgelegt.

### Schritte

1. **Source Namespace Owner:** Erstellen Sie das PVC (pvc1) im Quell-Namespace, der die Erlaubnis gibt, mit dem Ziel-Namespace zu teilen (namespace2) Mit dem shareToNamespace Anmerkung:

```
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 erstellt das PV und das dazugehörige Backend-NFS-Storage-Volume.

- Sie können das PVC über eine durch Kommas getrennte Liste mehreren Namespaces freigeben. Beispiel: trident.netapp.io/shareToNamespace: namespace2, namespace3, namespace4.
- Sie können mit allen Namespaces freigeben \*. Beispiel: trident.netapp.io/shareToNamespace: \*
- Sie können das PVC so aktualisieren, dass es die enthält shareToNamespace Kommentare können jederzeit hinzugefügt werden.



2. **Clusteradministrator:** Stellen Sie sicher, dass das richtige RBAC vorhanden ist, um dem Besitzer des Zielnamespace die Berechtigung zum Erstellen des TridentVolumeReference CR im Zielnamespace zu erteilen.
3. **Zielgebietes-Namespace-Eigentümer:** Erstellen Sie ein TridentVolumeReference CR im Ziel-Namespace, der sich auf den Quell-Namespace bezieht pvc1.

```

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

```

4. **Eigentümer des Ziel-Namespace:** Erstellen Sie ein PVC (pvc2) Im Ziel-Namespace (namespace2) Mit dem shareFromPVC Anmerkung zur Angabe der Quelle PVC.

```

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

```



Die Größe der Ziel-PVC muss kleiner oder gleich der Quelle PVC sein.

## Ergebnisse

Trident liest die shareFromPVC Annotation auf der Ziel-PVC und erstellt das Ziel-PV als ein untergeordnetes Volume ohne eigene Speicherressource, die auf das Quell-PV verweist und die Quell-PV-Speicherressource gemeinsam nutzt. Die Ziel-PVC und das PV erscheinen wie normal gebunden.

## Löschen eines freigegebenen Volumes

Sie können ein Volume löschen, das über mehrere Namespaces hinweg gemeinsam genutzt wird. Trident entfernt den Zugriff auf das Volume im Quell-Namespace und hat auch Zugriff auf andere Namespaces, die das Volume gemeinsam nutzen. Wenn alle Namespaces, die auf das Volume verweisen, entfernt werden, löscht Trident das Volume.

## Nutzung `tridentctl get` Zum Abfragen von untergeordneten Volumes

Verwenden der `tridentctl` Das Dienstprogramm kann ausgeführt werden `get` Befehl zum Abrufen untergeordneter Volumes. Weitere Informationen finden Sie unter Link:../Trident-

Usage:

```
tridentctl get [option]
```

Markierungen:

- ` -h, --help: Hilfe für Volumen.
- --parentOfSubordinate string: Abfrage auf untergeordnetes Quellvolumen begrenzen.
- --subordinateOf string: Abfrage auf Untergabe beschränken.

## Einschränkungen

- Trident kann nicht verhindern, dass Zielnamespaces auf das gemeinsam genutzte Volume schreiben. Sie sollten Dateisperren oder andere Prozesse verwenden, um das Überschreiben von gemeinsam genutzten Volume-Daten zu verhindern.
- Sie können den Zugriff auf die Quelle PVC nicht widerrufen, indem Sie die entfernen shareToNamespace Oder shareFromNamespace Anmerkungen oder Löschen des TridentVolumeReference CR. Um den Zugriff zu widerrufen, müssen Sie das untergeordnete PVC löschen.
- Snapshots, Klonen und Spiegelungen sind auf untergeordneten Volumes nicht möglich.

## Finden Sie weitere Informationen

Weitere Informationen zum Namespace-übergreifenden Volume-Zugriff:

- Besuchen Sie "[Teilen von Volumes zwischen Namespaces: Sagen Sie hallo für Namespace-übergreifenden Volume-Zugriff](#)".
- Sehen Sie sich die Demo an "[NetAppTV](#)".

## Volumes können in Namespaces geklont werden

Mit Trident können Sie neue Volumes unter Verwendung vorhandener Volumes oder Volume-Snapshots aus einem anderen Namespace im selben Kubernetes-Cluster erstellen.

## Voraussetzungen

Stellen Sie vor dem Klonen von Volumes sicher, dass die Quell- und Ziel-Back-Ends vom gleichen Typ sind und dieselbe Storage-Klasse aufweisen.



Das Klonen über Namespaces hinweg wird nur für die ontap-san Und ontap-nas Speichertreiber. Schreibgeschützte Klonen werden nicht unterstützt.

## Schnellstart

Die Einrichtung von Volume-Klonen ist in wenigen Schritten möglich.

1

### Konfigurieren Sie die Quell-PVC zum Klonen des Volume

Der Eigentümer des Quell-Namespace erteilt die Berechtigung, auf die Daten im Quell-PVC zuzugreifen.

2

### Berechtigung zum Erstellen eines CR im Ziel-Namespace gewähren

Der Clusteradministrator erteilt dem Eigentümer des Ziel-Namespace die Berechtigung, das TridentVolumeReference CR zu erstellen.

3

### Erstellen Sie im Ziel-Namespace tridentVolumeReference

Der Eigentümer des Ziel-Namespace erstellt das TridentVolumeReference CR, um sich auf das Quell-PVC zu beziehen.

4

### Erstellen Sie die Klon-PVC im Ziel-Namespace

Der Eigentümer des Ziel-Namespace erstellt die PVC zum Klonen der PVC aus dem Quell-Namespace.

## Konfigurieren Sie die Namensräume für Quelle und Ziel

Um die Sicherheit zu gewährleisten, müssen Volumes über Namespaces hinweg gemeinsam genutzt werden. Der Eigentümer des Quell-Namespace, der Cluster-Administrator und der Eigentümer des Ziel-Namespace müssen entsprechende Maßnahmen ergreifen. In jedem Schritt wird die Benutzerrolle festgelegt.

### Schritte

1. **Source Namespace Owner:** Erstellen Sie die PVC (pvc1) im source Namespace (namespace1), die die Erlaubnis erteilt, mit dem Ziel-Namespace zu teilen (namespace2) mit der `cloneToNamespace` Annotation.

```
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 erstellt das PV und das zugehörige Back-End Storage Volume.



- Sie können das PVC über eine durch Kommas getrennte Liste mehreren Namespaces freigeben. `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4` Beispiel: .
- Mit können Sie alle Namespaces teilen \*. Beispiel: trident.netapp.io/cloneToNamespace: \*
- Sie können die PVC so aktualisieren, dass die Anmerkung jederzeit enthalten cloneToNamespace ist.

2. **Clusteradministrator:** Stellen Sie sicher, dass die richtige RBAC vorhanden ist, um dem Besitzer des Zielnamespace die Berechtigung zum Erstellen des TridentVolumeReference CR im Zielnamespace zu erteilen.(namespace2 ).

3. **Zielgebietes-Namespace-Eigentümer:** Erstellen Sie ein TridentVolumeReference CR im Ziel- Namespace, der sich auf den Quell-Namespace bezieht pvc1.

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

4. **Destination Namespace Owner:** Erstellen Sie eine PVC (pvc2) im Destination Namespace (namespace2) Verwenden Sie cloneFromPVC die oder cloneFromSnapshot, und cloneFromNamespace Anmerkungen, um die Quell-PVC zu kennzeichnen.

```
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
```

## Einschränkungen

- Für PVCs, die über ONTAP-nas-Economy-Treiber bereitgestellt werden, werden schreibgeschützte Klone nicht unterstützt.

## Replizieren Sie Volumes mit SnapMirror

Trident unterstützt Spiegelungsbeziehungen zwischen einem Quell-Volume auf einem Cluster und dem Ziel-Volume auf dem Peering-Cluster, damit Daten für Disaster Recovery repliziert werden. Sie können eine benutzerdefinierte Ressourcendefinition (CRD) mit Namespace, genannt Trident Mirror Relationship (TMR), verwenden, um die folgenden Vorgänge auszuführen:

- Erstellen von Spiegelbeziehungen zwischen Volumes (VES)
- Entfernen Sie Spiegelungsbeziehungen zwischen Volumes
- Brechen Sie die Spiegelbeziehungen auf
- Bewerben des sekundären Volumes bei Ausfällen (Failover)
- Verlustfreie Transition von Applikationen von Cluster zu Cluster (während geplanter Failover oder Migrationen)

## Replikationsvoraussetzungen

Stellen Sie sicher, dass die folgenden Voraussetzungen erfüllt sind, bevor Sie beginnen:

### ONTAP Cluster

- **Trident:** Trident Version 22.10 oder höher muss sowohl auf den Quell- als auch auf den Ziel-Kubernetes-Clustern existieren, die ONTAP als Backend nutzen.
- **Lizenzen:** Asynchrone Lizenzen von ONTAP SnapMirror, die das Datensicherungspaket verwenden, müssen sowohl auf den Quell- als auch auf den Ziel-ONTAP-Clustern aktiviert sein. Weitere Informationen finden Sie unter "[Übersicht über die SnapMirror Lizenzierung in ONTAP](#)" .

Ab ONTAP 9.10.1 werden alle Lizenzen als NetApp-Lizenzdatei (NLF) bereitgestellt. Dabei handelt es sich um eine einzelne Datei, die mehrere Funktionen ermöglicht. Weitere Informationen finden Sie unter "[In ONTAP One enthaltene Lizenzen](#)" .



Es wird nur der asynchrone Schutz von SnapMirror unterstützt.

### Peering

- **Cluster und SVM:** Die ONTAP Speicher-Back-Ends müssen aktiviert werden. Weitere Informationen finden Sie unter "[Übersicht über Cluster- und SVM-Peering](#)" .



Vergewissern Sie sich, dass die in der Replizierungsbeziehung zwischen zwei ONTAP-Clustern verwendeten SVM-Namen eindeutig sind.

- **Trident und SVM:** Die Peered Remote SVMs müssen Trident auf dem Ziel-Cluster zur Verfügung stehen.

### Unterstützte Treiber

NetApp Trident unterstützt die Volume-Replikation mit der NetApp SnapMirror-Technologie unter Verwendung

von Speicherklassen, die von den folgenden Treibern unterstützt werden: **ontap-nas : NFS** ontap-san : iSCSI **ontap-san : FC** ontap-san : NVMe/TCP (erfordert mindestens ONTAP-Version 9.15.1)



Die Volume-Replikation mit SnapMirror wird für ASA R2-Systeme nicht unterstützt.  
Informationen zu ASA R2-Systemen finden Sie unter "["Erfahren Sie mehr über ASA r2 Storage-Systeme"](#)".

## Erstellen Sie eine gespiegelte PVC

Führen Sie die folgenden Schritte aus, und verwenden Sie die CRD-Beispiele, um eine Spiegelungsbeziehung zwischen primären und sekundären Volumes zu erstellen.

### Schritte

1. Führen Sie auf dem primären Kubernetes-Cluster die folgenden Schritte aus:
  - a. Erstellen Sie ein StorageClass-Objekt mit dem `trident.netapp.io/replication: true` Parameter.

#### Beispiel

```
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. PVC mit zuvor erstellter StorageClass erstellen.

#### Beispiel

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

- c. Erstellen Sie eine MirrorRelation CR mit lokalen Informationen.

## Beispiel

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

Trident ruft die internen Informationen für das Volume und den aktuellen DP-Status des Volumes ab und füllt dann das Statusfeld der MirrorRelation aus.

- d. Holen Sie sich den TridentMirrorRelationship CR, um den internen Namen und die SVM der PVC zu erhalten.

```
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. Führen Sie auf dem sekundären Kubernetes-Cluster die folgenden Schritte aus:

- a. Erstellen Sie eine StorageClass mit dem Parameter `trident.netapp.io/replication: true`.

## Beispiel

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

- b. Erstellen Sie eine MirrorRelationship-CR mit Ziel- und Quellinformationen.

## Beispiel

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

Trident erstellt eine SnapMirror Beziehung mit dem Namen der konfigurierten Beziehungsrichtlinie (oder dem Standard für ONTAP) und initialisiert diesen.

- c. PVC mit zuvor erstellter StorageClass erstellen, um als sekundäres Ziel zu fungieren (SnapMirror Ziel).

## Beispiel

```
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 überprüft die CRD der tridentMirrorRelationship und erstellt das Volume nicht, wenn die

Beziehung nicht vorhanden ist. Wenn die Beziehung besteht, stellt Trident sicher, dass die neue FlexVol volume auf einer SVM platziert wird, die mit der Remote-SVM, die in MirrorRelation definiert ist, verbunden ist.

## Volume-Replikationsstatus

Eine Trident Mirror-Beziehung (TMR) ist eine CRD, die ein Ende einer Replizierungsbeziehung zwischen PVCs darstellt. Das Ziel-TMR verfügt über einen Status, der Trident den gewünschten Status angibt. Das Ziel-TMR hat die folgenden Zustände:

- **Etabliert:** Die lokale PVC ist das Zielvolumen einer Spiegelbeziehung, und das ist eine neue Beziehung.
- **Befördert:** Die lokale PVC ist ReadWrite und montierbar, ohne dass aktuell eine Spiegelbeziehung besteht.
- **Wiederhergestellt:** Die lokale PVC ist das Zielvolumen einer Spiegelbeziehung und war zuvor auch in dieser Spiegelbeziehung.
  - Der neu eingerichtete Status muss verwendet werden, wenn das Ziel-Volume jemals in einer Beziehung zum Quell-Volume stand, da es den Inhalt des Ziel-Volume überschreibt.
  - Der neu eingerichtete Status schlägt fehl, wenn das Volume zuvor nicht in einer Beziehung zur Quelle stand.

## Fördern Sie die sekundäre PVC während eines ungeplanten Failover

Führen Sie den folgenden Schritt auf dem sekundären Kubernetes-Cluster aus:

- Aktualisieren Sie das Feld `spec.State` von `TridentMirrorRelationship` auf `promoted`.

## Fördern Sie die sekundäre PVC während eines geplanten Failover

Führen Sie während eines geplanten Failover (Migration) die folgenden Schritte durch, um die sekundäre PVC hochzustufen:

### Schritte

1. Erstellen Sie auf dem primären Kubernetes-Cluster einen Snapshot der PVC und warten Sie, bis der Snapshot erstellt wurde.
2. Erstellen Sie auf dem primären Kubernetes-Cluster `SnapshotInfo` CR, um interne Details zu erhalten.

### Beispiel

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

3. Aktualisieren Sie im sekundären Kubernetes-Cluster das Feld `spec.State` des `tridentMirrorRelationship` CR auf `promoted` und `spec.promotedSnapshotHandle` als InternalName des Snapshots.
4. Bestätigen Sie auf sekundärem Kubernetes-Cluster den Status (Feld `Status.State`) von

TridentMirrorRelationship auf hochgestuft.

## Stellen Sie nach einem Failover eine gespiegelte Beziehung wieder her

Wählen Sie vor dem Wiederherstellen einer Spiegelbeziehung die Seite aus, die Sie als neuen primären festlegen möchten.

### Schritte

1. Stellen Sie auf dem sekundären Kubernetes-Cluster sicher, dass die Werte für das Feld `spec.remoteVolumeHandle` auf dem TridentMirrorRelationship aktualisiert werden.
2. Aktualisieren Sie im sekundären Kubernetes-Cluster das Feld `spec.mirror` von TridentMirrorRelationship auf `reestablished`.

## Zusätzliche Vorgänge

Trident unterstützt folgende Vorgänge auf primären und sekundären Volumes:

### Replizieren der primären PVC auf eine neue sekundäre PVC

Stellen Sie sicher, dass Sie bereits über eine primäre PVC und eine sekundäre PVC verfügen.

### Schritte

1. Löschen Sie die CRDs PersistentVolumeClaim und TridentMirrorRelationship aus dem eingerichteten sekundären Cluster (Ziel).
2. Löschen Sie die CRD für TridentMirrorRelationship aus dem primären (Quell-) Cluster.
3. Erstellen Sie eine neue TridentMirrorRelationship CRD auf dem primären (Quell-) Cluster für die neue sekundäre (Ziel-) PVC, die Sie einrichten möchten.

### Ändern der Größe einer gespiegelten, primären oder sekundären PVC

Die PVC-Größe kann wie gewohnt geändert werden. ONTAP erweitert automatisch alle Zielflhxole, wenn die Datenmenge die aktuelle Größe überschreitet.

### Entfernen Sie die Replikation aus einer PVC

Um die Replikation zu entfernen, führen Sie einen der folgenden Vorgänge auf dem aktuellen sekundären Volume aus:

- Löschen Sie MirrorRelation auf der sekundären PVC. Dadurch wird die Replikationsbeziehung unterbrochen.
- Oder aktualisieren Sie das Feld `spec.State` auf `promoted`.

### Löschen einer PVC (die zuvor gespiegelt wurde)

Trident prüft, ob replizierte VES vorhanden sind, und gibt die Replizierungsbeziehung frei, bevor das Volume gelöscht werden soll.

### Löschen eines TMR

Das Löschen eines TMR auf einer Seite einer gespiegelten Beziehung führt dazu, dass der verbleibende TMR in den Status „`promoted`“ übergeht, bevor Trident den Löschkvorgang abgeschlossen hat. Wenn der für den Löschkvorgang ausgewählte TMR bereits den Status `heraufgestuft` hat, gibt es keine bestehende

Spiegelbeziehung und der TMR wird entfernt und Trident wird die lokale PVC auf *ReadWrite* hochstufen. Durch dieses Löschen werden SnapMirror Metadaten für das lokale Volume in ONTAP freigegeben. Wenn dieses Volume in Zukunft in einer Spiegelbeziehung verwendet wird, muss es beim Erstellen der neuen Spiegelbeziehung ein neues TMR mit einem *established* Volume-Replikationsstatus verwenden.

## Aktualisieren Sie Spiegelbeziehungen, wenn ONTAP online ist

Spiegelbeziehungen können jederzeit nach ihrer Einrichtung aktualisiert werden. Sie können die Felder oder verwenden `state: promoted` `state: reestablished`, um die Beziehungen zu aktualisieren. Wenn Sie ein Zielvolume auf ein reguläres ReadWrite-Volume heraufstufen, können Sie `promotedSnapshotHandle` verwenden, um einen bestimmten Snapshot anzugeben, auf dem das aktuelle Volume wiederhergestellt werden soll.

## Aktualisieren Sie Spiegelbeziehungen, wenn ONTAP offline ist

Sie können ein CRD verwenden, um ein SnapMirror-Update durchzuführen, ohne dass Trident über eine direkte Verbindung zum ONTAP-Cluster verfügt. Im folgenden Beispielformat finden Sie das `TridentActionMirrorUpdate`:

### Beispiel

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

`status.state` Gibt den Status von `TridentActionMirrorUpdate` CRD wieder. Es kann einen Wert von `suceeded`, `in progress` oder `failed` annehmen.

## Verwenden Sie die CSI-Topologie

Trident kann selektiv Volumes erstellen und an Nodes in einem Kubernetes-Cluster anhängen, indem Sie die verwenden "[Funktion CSI Topology](#)".

### Überblick

Mithilfe der CSI Topology-Funktion kann der Zugriff auf Volumes auf einen Teil von Nodes basierend auf Regionen und Verfügbarkeitszonen begrenzt werden. Cloud-Provider ermöglichen Kubernetes-Administratoren inzwischen das Erstellen von Nodes, die zonenbasiert sind. Die Nodes können sich in verschiedenen Verfügbarkeitszonen innerhalb einer Region oder über verschiedene Regionen hinweg befinden. Um die Bereitstellung von Volumes für Workloads in einer Architektur mit mehreren Zonen zu vereinfachen, verwendet Trident die CSI-Topologie.



Erfahren Sie mehr über die Funktion CSI Topology ["Hier"](#).

Kubernetes bietet zwei unterschiedliche Modi für die Volume-Bindung:

- Mit `VolumeBindingMode Set to Immediate` erzeugt Trident das Volumen ohne jegliche Topologiewahrnehmung. Die Volume-Bindung und die dynamische Bereitstellung werden bei der Erstellung des PVC behandelt. Dies ist die Standardeinstellung `VolumeBindingMode` und eignet sich für Cluster, die keine Topologieeinschränkungen erzwingen. Persistente Volumes werden erstellt, ohne von den Planungsanforderungen des anfragenden Pods abhängig zu sein.
- Mit `VolumeBindingMode Set to WaitForFirstConsumer`, Die Erstellung und Bindung eines Persistent Volume für ein PVC wird verzögert, bis ein Pod, der die PVC verwendet, geplant und erstellt wird. Auf diese Weise werden Volumes erstellt, um Planungseinschränkungen zu erfüllen, die durch Topologieanforderungen durchgesetzt werden.



Der `WaitForFirstConsumer` Für den Bindungsmodus sind keine Topologiebeschriftungen erforderlich. Diese kann unabhängig von der CSI Topology Funktion verwendet werden.

## Was Sie benötigen

Für die Verwendung von CSI Topology benötigen Sie Folgendes:

- Einen Kubernetes-Cluster mit einem "[Unterstützte Kubernetes-Version](#)"

```
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"}
```

- Nodes im Cluster sollten über Labels verfügen, die Topologiebewusstsein und `topology.kubernetes.io/zone` einführen(`topology.kubernetes.io/region`). Diese Bezeichnungen **sollten auf Knoten im Cluster** vorhanden sein, bevor Trident installiert ist, damit Trident topologiefähig ist.

```

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"}]

```

## Schritt 1: Erstellen Sie ein Topologieorientiertes Backend

Trident Storage-Back-Ends können so entworfen werden, dass sie Volumes selektiv basierend auf Verfügbarkeitszonen bereitstellen. Jedes Backend kann einen optionalen Block enthalten `supportedTopologies`, der eine Liste der unterstützten Zonen und Regionen darstellt. Bei `StorageClasses`, die ein solches Backend nutzen, wird ein Volume nur erstellt, wenn es von einer Applikation angefordert wird, die in einer unterstützten Region/Zone geplant ist.

Hier ist eine Beispiel-Backend-Definition:

## 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 Wird verwendet, um eine Liste von Regionen und Zonen pro Backend bereitzustellen. Diese Regionen und Zonen stellen die Liste der zulässigen Werte dar, die in einer StorageClass bereitgestellt werden können. Bei StorageClasses, die eine Teilmenge der Regionen und Zonen enthalten, die in einem Back-End bereitgestellt werden, erstellt Trident auf dem Back-End ein Volume.

Sie können definieren `supportedTopologies` Auch pro Storagepool. Das folgende Beispiel zeigt:

```
---
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 diesem Beispiel ist der `region` Und `zone` Etiketten stehen für die Position des Speicherpools. `topology.kubernetes.io/region` Und `topology.kubernetes.io/zone` Vorgeben, woher die Speicherpools verbraucht werden können.

## Schritt: Definition von StorageClasses, die sich der Topologie bewusst sind

Auf der Grundlage der Topologiebeschriftungen, die den Nodes im Cluster zur Verfügung gestellt werden, können StorageClasses so definiert werden, dass sie Topologieinformationen enthalten. So werden die Storage-Pools festgelegt, die als Kandidaten für PVC-Anfragen dienen, und die Untergruppe der Nodes, die die von Trident bereitgestellten Volumes nutzen können.

Das folgende Beispiel zeigt:

```

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

```

In der oben angegebenen StorageClass-Definition volumeBindingMode ist auf festgelegt WaitForFirstConsumer. VES, die mit dieser StorageClass angefordert werden, werden erst dann gehandelt, wenn sie in einem Pod referenziert werden. Und allowedTopologies stellt die zu verwendenden Zonen und Regionen bereit. Die netapp-san-us-east1 StorageClass erstellt VES auf dem san-backend-us-east1 oben definierten Back-End.

### Schritt 3: Erstellen und verwenden Sie ein PVC

Wenn die StorageClass erstellt und einem Backend zugeordnet wird, können Sie jetzt PVCs erstellen.

Siehe Beispiel spec Unten:

```

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

```

Das Erstellen eines PVC mithilfe dieses Manifests würde Folgendes zur Folge haben:

```

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

```

Verwenden Sie für Trident, ein Volume zu erstellen und es an die PVC zu binden, das in einem Pod verwendet wird. Das folgende Beispiel zeigt:

```

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

```

Diese PodSpec beauftragt Kubernetes, den Pod auf Nodes zu planen, die in vorhanden sind `us-east1`  
Wählen Sie einen beliebigen Knoten aus, der im vorhanden ist `us-east1-a` Oder `us-east1-b` Zonen:

Siehe die folgende Ausgabe:

```

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

```

## Aktualisieren Sie Back-Ends, um einzuschließen supportedTopologies

Vorhandene Back-Ends können mit einer Liste von aktualisiert werden supportedTopologies Wird verwendet tridentctl backend update. Dies wirkt sich nicht auf Volumes aus, die bereits bereitgestellt wurden und nur für nachfolgende VES verwendet werden.

## Weitere Informationen

- "[Management von Ressourcen für Container](#)"
- "[NodeSelector](#)"
- "[Affinität und Antiaffinität](#)"
- "[Tönungen und Tolerationen](#)"

## Arbeiten Sie mit Snapshots

Kubernetes Volume Snapshots von Persistent Volumes (PVs) ermöglichen zeitpunktgenaue Kopien von Volumes. Sie können einen Snapshot eines mit Trident erstellten Volumes erstellen, einen außerhalb von Trident erstellten Snapshot importieren, ein neues Volume aus einem vorhandenen Snapshot erstellen und Volume-Daten aus Snapshots wiederherstellen.

## Überblick

Volume-Snapshots werden unterstützt von `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files`, Und `google-cloud-netapp-volumes` Fahrer.

### Bevor Sie beginnen

Sie benötigen einen externen Snapshot-Controller und benutzerdefinierte Ressourcendefinitionen (CRDs), um mit Snapshots arbeiten zu können. Dies ist die Aufgabe des Kubernetes Orchestrator (z. B. Kubeadm, GKE, OpenShift).

Wenn die Kubernetes-Distribution den Snapshot-Controller und die CRDs nicht enthält, lesen Sie [Stellen Sie einen Volume-Snapshot-Controller bereit](#).



Erstellen Sie keinen Snapshot Controller, wenn Sie On-Demand Volume Snapshots in einer GKE-Umgebung erstellen. GKE verwendet einen integrierten, versteckten Snapshot-Controller.

## Erstellen eines Volume-Snapshots

### Schritte

1. Erstellen Sie ein `VolumeSnapshotClass`. Weitere Informationen finden Sie unter "[VolumeSnapshotKlasse](#)".
  - Der `driver` verweist auf den Trident-CSI-Treiber.
  - `deletionPolicy` Kann sein `Delete` Oder `Retain`. Wenn eingestellt auf `Retain`, Der zugrunde liegende physische Snapshot auf dem Storage-Cluster wird auch dann beibehalten, wenn der `VolumeSnapshot` Objekt wurde gelöscht.

### Beispiel

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

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

2. Erstellen Sie einen Snapshot einer vorhandenen PVC.

### Beispiele

- In diesem Beispiel wird ein Snapshot eines vorhandenen PVC erstellt.

```
cat snap.yaml
```

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

- In diesem Beispiel wird ein Volume-Snapshot-Objekt für eine PVC mit dem Namen erstellt `pvc1` Der Name des Snapshots lautet `pvc1-snap`. Ein `VolumeSnapshot` ist analog zu einem PVC und einem zugeordnet `VolumeSnapshotContent` Objekt, das den tatsächlichen Snapshot darstellt.

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

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

- Sie können den identifizieren VolumeSnapshotContent Objekt für das pvc1-snap VolumeSnapshot wird beschrieben. Der Snapshot Content Name Identifiziert das VolumeSnapshotContent-Objekt, das diesen Snapshot bereitstellt. Der Ready To Use Parameter gibt an, dass der Snapshot zum Erstellen einer neuen PVC verwendet werden kann.

```
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:   PersistentVolumeClaim
    Kind:        PersistentVolumeClaim
    Name:         pvc1
  Status:
    Creation Time: 2019-06-26T15:27:29Z
    Ready To Use:  true
    Restore Size:  3Gi
...
...
```

## Erstellen Sie eine PVC aus einem Volume-Snapshot

Verwenden Sie können dataSource So erstellen Sie eine PVC mit einem VolumeSnapshot namens <pvc-name> Als Quelle der Daten. Nachdem die PVC erstellt wurde, kann sie an einem Pod befestigt und wie jedes andere PVC verwendet werden.

 Die PVC wird im selben Backend wie das Quell-Volume erstellt. Siehe "[KB: Die Erstellung einer PVC aus einem Trident PVC-Snapshot kann nicht in einem alternativen Backend erstellt werden](#)".

Im folgenden Beispiel wird die PVC mit erstellt pvc1-snap Als Datenquelle speichern.

```
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

```

## Importieren Sie einen Volume-Snapshot

Trident unterstützt das, damit der "Vorab bereitgestellter Snapshot-Prozess von Kubernetes" Clusteradministrator ein Objekt erstellen und Snapshots importieren kann VolumeSnapshotContent, die außerhalb von Trident erstellt wurden.

### Bevor Sie beginnen

Trident muss das übergeordnete Volume des Snapshots erstellt oder importiert haben.

### Schritte

1. **Cluster admin:** Erstellen Sie ein VolumeSnapshotContent Objekt, das auf den Back-End-Snapshot verweist. Dadurch wird der Snapshot Workflow in Trident gestartet.
  - Geben Sie den Namen des Back-End-Snapshots in an annotations Als trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">.
  - Geben Sie <name-of-parent-volume-in-trident>/<volume-snapshot-content-name> in an snapshotHandle. Dies ist die einzige Information, die Trident vom externen Snapshotter im Aufruf zur Verfügung gestellt ListSnapshots wird.



Der <volumeSnapshotContentName> Aufgrund von Einschränkungen bei der CR-Benennung kann der Name des Back-End-Snapshots nicht immer übereinstimmen.

### Beispiel

Im folgenden Beispiel wird ein erstellt VolumeSnapshotContent Objekt, das auf Back-End-Snapshot verweist 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:** Erstellen Sie das VolumeSnapshot CR, der auf den verweist VolumeSnapshotContent Objekt: Dadurch wird der Zugriff auf die Verwendung des angefordert VolumeSnapshot In einem bestimmten Namespace.

### Beispiel

Im folgenden Beispiel wird ein erstellt VolumeSnapshot CR benannt import-snap Die auf die verweisen VolumeSnapshotContent Genannt 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. **Interne Verarbeitung (keine Aktion erforderlich):** der externe Schnapper erkennt das neu erstellte VolumeSnapshotContent und führt den ListSnapshots Aufruf aus. Trident erstellt die TridentSnapshot.

- Der externe Schnapper legt den fest VolumeSnapshotContent Bis readyToUse Und das VolumeSnapshot Bis true.
- Trident kehrt zurück readyToUse=true.

4. **Jeder Benutzer:** Erstellen Sie eine PersistentVolumeClaim Um auf das neue zu verweisen VolumeSnapshot, Wo der spec.dataSource (Oder spec.dataSourceRef) Name ist der

VolumeSnapshot Name:

### Beispiel

Im folgenden Beispiel wird eine PVC erstellt, die auf den verweist volumeSnapshot Genannt 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
```

## Stellen Sie Volume-Daten mithilfe von Snapshots wieder her

Das Snapshot-Verzeichnis ist standardmäßig ausgeblendet, um die maximale Kompatibilität von Volumes zu ermöglichen, die über bereitgestellt werden ontap-nas Und ontap-nas-economy Treiber. Aktivieren Sie die .snapshot Verzeichnis, um Daten von Snapshots direkt wiederherzustellen.

Verwenden Sie die ONTAP-CLI zur Wiederherstellung eines Volume-Snapshots, um einen in einem früheren Snapshot aufgezeichneten Zustand wiederherzustellen.

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



Wenn Sie eine Snapshot-Kopie wiederherstellen, wird die vorhandene Volume-Konfiguration überschrieben. Änderungen an den Volume-Daten nach der Erstellung der Snapshot Kopie gehen verloren.

## In-Place-Volume-Wiederherstellung aus einem Snapshot

Trident ermöglicht mithilfe des CR-Systems (TASR) eine schnelle Wiederherstellung von in-Place-Volumes aus einem Snapshot TridentActionSnapshotRestore. Dieser CR fungiert als eine zwingend notwendige Kubernetes-Aktion und bleibt nach Abschluss des Vorgangs nicht erhalten.

Trident unterstützt die Wiederherstellung von Snapshots auf dem ontap-san , ontap-san-economy , ontap-nas , ontap-nas-flexgroup , azure-netapp-files , google-cloud-netapp-volumes , Und

solidfire-san Fahrer.

## Bevor Sie beginnen

Sie müssen über einen gebundenen PVC-Snapshot und einen verfügbaren Volume-Snapshot verfügen.

- Vergewissern Sie sich, dass der PVC-Status gebunden ist.

```
kubectl get pvc
```

- Überprüfen Sie, ob der Volume-Snapshot einsatzbereit ist.

```
kubectl get vs
```

## Schritte

1. Erstellen Sie den TASR CR. In diesem Beispiel wird ein CR für PVC und Volume-Snapshot erstellt pvc1-pvc1-snapshot.



Der TASR CR muss sich in einem Namensraum befinden, in dem PVC und VS vorhanden sind.

```
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. Wenden Sie den CR an, um ihn aus dem Snapshot wiederherzustellen. Dieses Beispiel wird aus Snapshot wiederhergestellt pvc1.

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

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

## Ergebnisse

Trident stellt die Daten aus dem Snapshot wieder her. Sie können den Wiederherstellungsstatus von

Snapshots überprüfen:

```
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: ""
```

-  • In den meisten Fällen versucht Trident den Vorgang bei einem Ausfall nicht automatisch erneut. Sie müssen den Vorgang erneut ausführen.
- Kubernetes-Benutzer ohne Administratorzugriff müssen möglicherweise vom Administrator zum Erstellen eines TASR CR in ihrem Applikations-Namespace erhalten.

## Löschen Sie ein PV mit den zugehörigen Snapshots

Beim Löschen eines persistenten Volumes mit zugeordneten Snapshots wird das entsprechende Trident-Volume auf den „Löschstatus“ aktualisiert. Entfernen Sie die Volume-Snapshots, um das Trident-Volume zu löschen.

## Stellen Sie einen Volume-Snapshot-Controller bereit

Wenn Ihre Kubernetes-Distribution den Snapshot-Controller und CRDs nicht enthält, können Sie sie wie folgt bereitstellen.

### Schritte

1. Erstellen von Volume Snapshot-CRDs.

```
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. Erstellen Sie den Snapshot-Controller.

```
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
```



Öffnen Sie bei Bedarf `deploy/kubernetes/snapshot-controller/rbac-
snapshot-controller.yaml` Und Aktualisierung namespace In Ihren Namespace.

## Weiterführende Links

- "[Volume Snapshots](#)"
- "[VolumeSnapshotKlasse](#)"

## Arbeiten mit Volumegruppen-Snapshots

Kubernetes-Volumegruppen-Snapshots von persistenten Volumes (PVs). NetApp Trident ermöglicht die Erstellung von Snapshots mehrerer Volumes (einer Gruppe von Volume-Snapshots). Dieser Volumegruppen-Snapshot stellt Kopien mehrerer Volumes dar, die zum gleichen Zeitpunkt erstellt wurden.



VolumeGroupSnapshot ist eine Beta-Funktion in Kubernetes mit Beta-APIs. Für VolumeGroupSnapshot ist mindestens Kubernetes 1.32 erforderlich.

## Erstellen von Volumegruppen-Snapshots

Die Snapshot-Funktion für Volumengruppen wird von den folgenden Speichertreibern unterstützt:

- `ontap-san` Treiber - nur für die iSCSI- und FC-Protokolle, nicht für das NVMe/TCP-Protokoll.
- `ontap-san-economy` - nur für das iSCSI-Protokoll.
- `ontap-nas`



Die Erstellung von Volume-Group-Snapshots wird für NetApp ASA r2- oder AFX-Speichersysteme nicht unterstützt.

### Bevor Sie beginnen

- Stellen Sie sicher, dass Ihre Kubernetes-Version K8s 1.32 oder höher ist.
- Sie benötigen einen externen Snapshot-Controller und benutzerdefinierte Ressourcendefinitionen (CRDs), um mit Snapshots arbeiten zu können. Dies ist die Aufgabe des Kubernetes Orchestrator (z. B. Kubeadm, GKE, OpenShift).

Wenn Ihre Kubernetes-Distribution den externen Snapshot-Controller und die CRDs nicht enthält, lesen Sie [Stellen Sie einen Volume-Snapshot-Controller bereit](#).



Erstellen Sie keinen Snapshot-Controller, wenn Sie On-Demand-Volume-Gruppen-Snapshots in einer GKE-Umgebung erstellen. GKE verwendet einen integrierten, versteckten Snapshot-Controller.

- Legen Sie im Snapshot-Controller-YAML Folgendes fest: `CSIVolumeGroupSnapshot` Setzen Sie das Feature Gate auf „true“, um sicherzustellen, dass der Volumegruppen-Snapshot aktiviert ist.
- Erstellen Sie die erforderlichen Snapshot-Klassen für Volumegruppen, bevor Sie einen Snapshot für Volumegruppen erstellen.
- Stellen Sie sicher, dass sich alle PVCs/Volumes auf derselben SVM befinden, um VolumeGroupSnapshot erstellen zu können.

### Schritte

- Erstellen Sie eine `VolumeGroupSnapshotClass`, bevor Sie einen `VolumeGroupSnapshot` erstellen. Weitere Informationen finden Sie unter ["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
```

- Erstellen Sie PVCs mit den erforderlichen Bezeichnungen unter Verwendung vorhandener Speicherklassen oder fügen Sie diese Bezeichnungen zu vorhandenen PVCs hinzu.

Das folgende Beispiel erstellt den PVC mit pvc1-group-snap als Datenquelle und Bezeichnung consistentGroupSnapshot: groupA . Definieren Sie den Label-Schlüssel und den Wert entsprechend Ihren Anforderungen.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: scl-1
```

- Erstellen Sie einen VolumeGroupSnapshot mit demselben Label (consistentGroupSnapshot: groupA ) im PVC angegeben.

In diesem Beispiel wird ein Snapshot einer Volume-Gruppe erstellt:

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

## **Wiederherstellen von Volumedaten mithilfe eines Gruppen-Snapshots**

Sie können einzelne persistente Volumes mithilfe der einzelnen Snapshots wiederherstellen, die im Rahmen des Volume-Gruppen-Snapshots erstellt wurden. Der Volume-Gruppen-Snapshot kann nicht als Einheit wiederhergestellt werden.

Verwenden Sie die ONTAP-CLI zur Wiederherstellung eines Volume-Snapshots, um einen in einem früheren Snapshot aufgezeichneten Zustand wiederherzustellen.

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



Wenn Sie eine Snapshot-Kopie wiederherstellen, wird die vorhandene Volume-Konfiguration überschrieben. Änderungen an den Volume-Daten nach der Erstellung der Snapshot Kopie gehen verloren.

## In-Place-Volume-Wiederherstellung aus einem Snapshot

Trident ermöglicht mithilfe des CR-Systems (TASR) eine schnelle Wiederherstellung von in-Place-Volumes aus einem Snapshot TridentActionSnapshotRestore. Dieser CR fungiert als eine zwingend notwendige Kubernetes-Aktion und bleibt nach Abschluss des Vorgangs nicht erhalten.

Weitere Informationen finden Sie unter "[In-Place-Volume-Wiederherstellung aus einem Snapshot](#)".

## Löschen eines PV mit zugehörigen Gruppen-Snapshots

Beim Löschen eines Gruppenvolume-Snapshots:

- Sie können VolumeGroupSnapshots als Ganzes löschen, nicht jedoch einzelne Snapshots in der Gruppe.
- Wenn PersistentVolumes gelöscht werden, während ein Snapshot für dieses PersistentVolume vorhanden ist, versetzt Trident dieses Volume in den Status „Löschen“, da der Snapshot entfernt werden muss, bevor das Volume sicher entfernt werden kann.
- Wenn ein Klon mithilfe eines gruppierten Snapshots erstellt wurde und die Gruppe anschließend gelöscht werden soll, beginnt ein Split-on-Clone-Vorgang und die Gruppe kann erst gelöscht werden, wenn die Aufteilung abgeschlossen ist.

## Stellen Sie einen Volume-Snapshot-Controller bereit

Wenn Ihre Kubernetes-Distribution den Snapshot-Controller und CRDs nicht enthält, können Sie sie wie folgt bereitstellen.

### Schritte

1. Erstellen von Volume Snapshot-CRDs.

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

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

## 2. Erstellen Sie den Snapshot-Controller.

```
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
```



Öffnen Sie bei Bedarf `deploy/kubernetes/snapshot-controller/rbac-
snapshot-controller.yaml` Und Aktualisierung namespace In Ihren Namespace.

## Weiterführende Links

- "[VolumeGroupSnapshotClass](#)"
- "[Volume Snapshots](#)"

## **Copyright-Informationen**

Copyright © 2026 NetApp. Alle Rechte vorbehalten. Gedruckt in den USA. Dieses urheberrechtlich geschützte Dokument darf ohne die vorherige schriftliche Genehmigung des Urheberrechtsinhabers in keiner Form und durch keine Mittel – weder grafische noch elektronische oder mechanische, einschließlich Fotokopieren, Aufnehmen oder Speichern in einem elektronischen Abrufsystem – auch nicht in Teilen, vervielfältigt werden.

Software, die von urheberrechtlich geschütztem NetApp Material abgeleitet wird, unterliegt der folgenden Lizenz und dem folgenden Haftungsausschluss:

DIE VORLIEGENDE SOFTWARE WIRD IN DER VORLIEGENDEN FORM VON NETAPP ZUR VERFÜGUNG GESTELLT, D. H. OHNE JEGLICHE EXPLIZITE ODER IMPLIZITE GEWÄHRLEISTUNG, EINSCHLIESSLICH, JEDOCH NICHT BESCHRÄNKKT AUF DIE STILLSCHWEIGENDE GEWÄHRLEISTUNG DER MARKTGÄNGIGKEIT UND EIGNUNG FÜR EINEN BESTIMMTEN ZWECK, DIE HIERMIT AUSGESCHLOSSEN WERDEN. NETAPP ÜBERNIMMT KEINERLEI HAFTUNG FÜR DIREKTE, INDIREKTE, ZUFÄLLIGE, BESONDERE, BEISPIELHAFFE SCHÄDEN ODER FOLGESCHÄDEN (EINSCHLIESSLICH, JEDOCH NICHT BESCHRÄNKKT AUF DIE BESCHAFFUNG VON ERSATZWAREN ODER -DIENSTLEISTUNGEN, NUTZUNGS-, DATEN- ODER GEWINNVERLUSTE ODER UNTERBRECHUNG DES GESCHÄFTSBETRIEBS), UNABHÄNGIG DAVON, WIE SIE VERURSACHT WURDEN UND AUF WELCHER HAFTUNGSTHEORIE SIE BERUHEN, OB AUS VERTRAGLICH FESTGELEGTER HAFTUNG, VERSCHULDENSUNABHÄNGIGER HAFTUNG ODER DELIKTSHAFTUNG (EINSCHLIESSLICH FAHRLÄSSIGKEIT ODER AUF ANDEREM WEGE), DIE IN IRGENDERWEINER WEISE AUS DER NUTZUNG DIESER SOFTWARE RESULTIEREN, SELBST WENN AUF DIE MÖGLICHKEIT DERARTIGER SCHÄDEN HINGEWIESEN WURDE.

NetApp behält sich das Recht vor, die hierin beschriebenen Produkte jederzeit und ohne Vorankündigung zu ändern. NetApp übernimmt keine Verantwortung oder Haftung, die sich aus der Verwendung der hier beschriebenen Produkte ergibt, es sei denn, NetApp hat dem ausdrücklich in schriftlicher Form zugestimmt. Die Verwendung oder der Erwerb dieses Produkts stellt keine Lizenzierung im Rahmen eines Patentrechts, Markenrechts oder eines anderen Rechts an geistigem Eigentum von NetApp dar.

Das in diesem Dokument beschriebene Produkt kann durch ein oder mehrere US-amerikanische Patente, ausländische Patente oder anhängige Patentanmeldungen geschützt sein.

**ERLÄUTERUNG ZU „RESTRICTED RIGHTS“:** Nutzung, Vervielfältigung oder Offenlegung durch die US-Regierung unterliegt den Einschränkungen gemäß Unterabschnitt (b)(3) der Klausel „Rights in Technical Data – Noncommercial Items“ in DFARS 252.227-7013 (Februar 2014) und FAR 52.227-19 (Dezember 2007).

Die hierin enthaltenen Daten beziehen sich auf ein kommerzielles Produkt und/oder einen kommerziellen Service (wie in FAR 2.101 definiert) und sind Eigentum von NetApp, Inc. Alle technischen Daten und die Computersoftware von NetApp, die unter diesem Vertrag bereitgestellt werden, sind gewerblicher Natur und wurden ausschließlich unter Verwendung privater Mittel entwickelt. Die US-Regierung besitzt eine nicht ausschließliche, nicht übertragbare, nicht unterlizenzierbare, weltweite, limitierte unwiderrufliche Lizenz zur Nutzung der Daten nur in Verbindung mit und zur Unterstützung des Vertrags der US-Regierung, unter dem die Daten bereitgestellt wurden. Sofern in den vorliegenden Bedingungen nicht anders angegeben, dürfen die Daten ohne vorherige schriftliche Genehmigung von NetApp, Inc. nicht verwendet, offengelegt, vervielfältigt, geändert, aufgeführt oder angezeigt werden. Die Lizenzrechte der US-Regierung für das US-Verteidigungsministerium sind auf die in DFARS-Klausel 252.227-7015(b) (Februar 2014) genannten Rechte beschränkt.

## **Markeninformationen**

NETAPP, das NETAPP Logo und die unter <http://www.netapp.com/TM> aufgeführten Marken sind Marken von NetApp, Inc. Andere Firmen und Produktnamen können Marken der jeweiligen Eigentümer sein.