Provisionierung und Management von Volumes

Astra Trident

NetApp
January 14, 2026

Inhalt

Provisionierung und Management von Volumes

Bereitstellen eines Volumes

Erstellen Sie ein PersistentVolume (PV) und 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

Ein "*PersistentVolume*" (PV) ist eine physische Speicherressource, die vom Clusteradministrator auf einem Kubernetes-Cluster bereitgestellt wird. Die "*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 das PV und die PVC erstellt haben, können Sie das Volume in einem Pod einbinden.

Beispielmanifeste

PersistentVolume-Beispielmanifest

Dieses Beispielmanifest zeigt ein Basis-PV von 10Gi, das mit StorageClass verknüpft ist `basic-csi`.

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

PersistentVolumeClaim-Beispielmanifeste

Diese Beispiele zeigen grundlegende PVC-Konfigurationsoptionen.

PVC mit RWO-Zugang

Dieses Beispiel zeigt ein einfaches PVC mit RWO-Zugriff, das mit einer StorageClass namens verknüpft 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 namens 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: pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

Grundlegende NVMe/TCP-Konfiguration

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

Erstellen Sie das PV und die PVC

Schritte

1. Erstellen Sie das PV.

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

2. Überprüfen Sie den PV-Status.

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

3. Erstellen Sie das PVC.

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

4. Überprüfen Sie den PVC-Status.

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

5. Mounten Sie das Volume in einem Pod.

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

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

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

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

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

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

Einzelheiten zur Interaktion von Storage-Klassen mit den `PersistentVolumeClaim` Parametern und zur Steuerung, wie Astra Trident Volumes provisioniert, finden Sie unter"Kubernetes und Trident Objekte".

Erweitern Sie Volumes

Astra Trident bietet Kubernetes-Benutzern die Möglichkeit, ihre Volumes nach Erstellung zu erweitern. Hier finden Sie Informationen zu den erforderlichen Konfigurationen zum erweitern von iSCSI- und NFS-Volumes.

Erweitern Sie ein iSCSI-Volume

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

> Die iSCSI-Volume-Erweiterung wird von den, `ontap-san-economy`- `solidfire-san`Treibern unterstützt `ontap-san` und erfordert Kubernetes 1.16 und höher.

Schritt: Storage Class für Volume-Erweiterung konfigurieren

Bearbeiten Sie die StorageClass-Definition, um das Feld auf `true` einzustellen `allowVolumeExpansion`.

```
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 diese, um den Parameter einzuschließen `allowVolumeExpansion`.

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

Bearbeiten Sie die PVC-Definition, und aktualisieren Sie den `spec.resources.requests.storage`, um die neu gewünschte Größe wiederzugeben, die größer sein muss als die ursprüngliche Größe.

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

Astra Trident erstellt ein persistentes Volume (PV) und verknüpft es mit dieser 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 an einen POD angeschlossen ist, erweitert Astra Trident das Volume auf dem Storage-Back-End, setzt das Gerät neu ein und vergrößert das Dateisystem neu.
- Bei dem Versuch, die Größe eines nicht angeschlossenen PV zu ändern, erweitert Astra Trident das Volume auf dem Storage-Backend. 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, der von 1Gi auf 2Gi erstellt wurde, zu ändern, bearbeiten Sie die PVC-Definition und aktualisieren Sie den `spec.resources.requests.storage` auf 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: Validierung der Erweiterung

Sie können die korrekte Ausführung der Erweiterung überprüfen, indem Sie die Größe der PVC, PV und des Astra 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

Astra Trident unterstützt Volume-Erweiterung für NFS PVS, die auf, `ontap-nas-economy`, `ontap-nas-flexgroup` `gcp-cvs` und `azure-netapp-files` Back-Ends bereitgestellt `ontap-nas` wurden.

Schritt: Storage Class für Volume-Erweiterung konfigurieren

Um die Größe eines NFS-PV zu ändern, muss der Administrator zuerst die Speicherklasse konfigurieren, um die Volume-Erweiterung zu ermöglichen, indem er das Feld auf `true`folgende Einstellung setzt `allowVolumeExpansion`:

```
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 und die Volume-Erweiterung zulassen. `kubectl edit storageclass`

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
```

Astra Trident sollte ein 20MiB NFS PV für diese PVC 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 20MiB-PV auf 1 gib zu ändern, bearbeiten Sie die PVC und setzen Sie `spec.resources.requests.storage` auf 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: Validierung der Erweiterung

Sie können die korrekte Größenänderung validieren, indem Sie die Größe des PVC, des PV und des Astra 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 Storage-Volumes mit importieren `tridentctl import`.

Überblick und Überlegungen

Ein Volume kann in Astra Trident importiert werden, 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.

- Astra Trident kann nur ONTAP Volumes vom Typ RW (Lese-/Schreibzugriff) importieren. Volumes im DP-Typ (Datensicherung) sind SnapMirror Ziel-Volumes. Sie sollten die Spiegelungsbeziehung unterbrechen, bevor Sie das Volume in Astra 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 Astra 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 Zurückgewinnungsrichtlinie ist zunächst im PV auf festgelegt `retain`. Nachdem Kubernetes die PVC und das PV erfolgreich bindet, wird die Zurückgewinnungsrichtlinie aktualisiert und an die Zurückgewinnungsrichtlinie der Storage-Klasse angepasst.
  - Wenn die Zurückgewinnungsrichtlinie der Speicherklasse lautet `delete`, wird das Speichervolume gelöscht, wenn das PV gelöscht wird.
- Astra Trident verwaltet standardmäßig die PVC und benennt die FlexVol und die LUN auf dem Backend um. Sie können das Flag übergeben `--no-manage`, um ein nicht verwaltetes Volume zu importieren. Wenn Sie verwenden `--no-manage`, führt Astra Trident keine zusätzlichen Operationen auf der PVC oder PV für den Lebenszyklus der Objekte aus. Das Speicher-Volume wird nicht gelöscht, wenn das PV gelöscht wird und andere Vorgänge wie Volume-Klon und Volume-Größe ebenfalls ignoriert werden.

  Diese Option ist nützlich, wenn Sie Kubernetes für Workloads in Containern verwenden möchten, aber ansonsten den Lebenszyklus des Storage Volumes außerhalb von Kubernetes managen 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 zum Importieren eines Volumes verwenden `tridentctl import`.

Schritte

1. Erstellen Sie die PVC-Datei (Persistent Volume Claim) (z. B. `pvc.yaml`), die zum Erstellen der PVC verwendet wird. Die PVC-Datei sollte , , `namespace` `accessModes` und `storageClassName` enthalten `name`. Optional können Sie in Ihrer PVC-Definition angeben `unixPermissions`.

   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
```

Verwenden Sie keine zusätzlichen Parameter wie den PV-Namen oder die Volume-Größe. Dies kann dazu führen, dass der Importbefehl fehlschlägt.

2. Verwenden Sie den `tridentctl import` Befehl, um den Namen des Astra Trident-Backends anzugeben, das das Volume enthält, sowie den Namen, der das Volume im Storage eindeutig identifiziert (z. B. ONTAP FlexVol, Element Volume oder Cloud Volumes Service-Pfad). Das `-f` Argument ist erforderlich, um den Pfad zur PVC-Datei anzugeben.

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

Beispiele

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

ONTAP NAS und ONTAP NAS FlexGroup

Astra Trident unterstützt den Volume-Import mithilfe der `ontap-nas` Treiber und `ontap-nas-flexgroup`.

- Der `ontap-nas-economy` Treiber kann qtrees nicht importieren und managen.
- Die `ontap-nas` und `ontap-nas-flexgroup`-Treiber erlauben keine doppelten Volume-Namen.

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

Beispiele für ONTAP NAS

Die folgende Darstellung zeigt ein Beispiel für ein verwaltetes Volume und einen nicht verwalteten Volume-Import.

Gemanagtes Volume

Das folgende Beispiel importiert ein Volume mit dem Namen `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 Astra Trident das Volume nicht um.

Im folgenden Beispiel werden Importe auf das `ontap_nas` Backend importiert `unmanaged_volume`:

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

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

ONTAP SAN

Astra Trident unterstützt den Volume-Import mithilfe des `ontap-san` Treibers. Der Import von Volumes wird mit dem Treiber nicht unterstützt `ontap-san-economy`.

Astra Trident kann ONTAP SAN FlexVols importieren, die eine einzige LUN enthalten. Dies ist mit dem Treiber konsistent `ontap-san`, der für jede PVC und eine LUN in der FlexVol eine FlexVol erstellt. Astra Trident importiert die FlexVol und ordnet sie der PVC-Definition zu.

Beispiele für ONTAP SAN

Die folgende Darstellung zeigt ein Beispiel für ein verwaltetes Volume und einen nicht verwalteten Volume-Import.

Gemanagtes Volume

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

Im folgenden Beispiel werden die auf dem Backend vorhandenen FlexVol `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    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

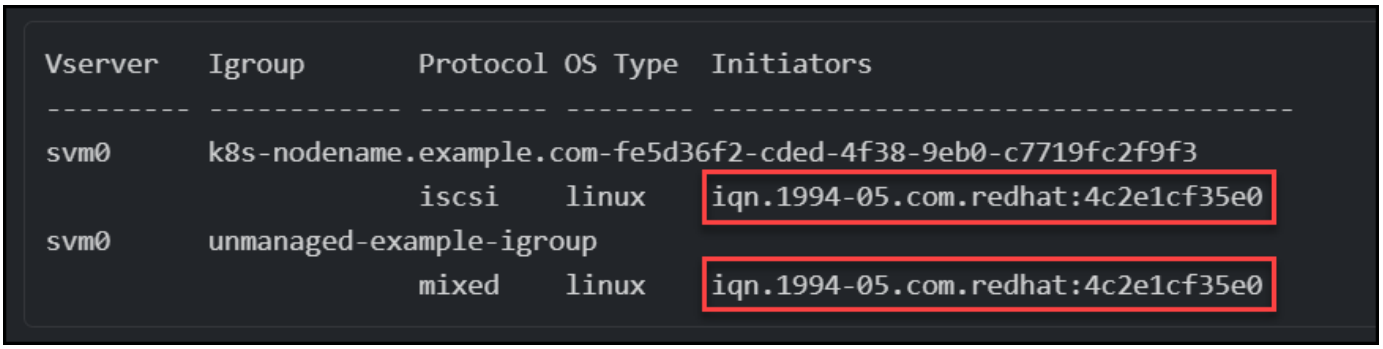
Nicht verwaltetes Volume

Im folgenden Beispiel werden Importe auf das `ontap_san` Backend importiert `unmanaged_example_volume`:

```
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.



Element

Astra Trident unterstützt die NetApp Element-Software sowie den NetApp HCI-Volume-Import über den `solidfire-san` Treiber.

> Der Elementtreiber unterstützt doppelte Volume-Namen. Astra 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.

Beispiel für ein Element

Das folgende Beispiel importiert ein `element-managed` Volume auf dem 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    |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
```

Google Cloud Platform

Astra Trident unterstützt den Volume-Import mithilfe des `gcp-cvs` Treibers.

Um ein Volume zu importieren, das von NetApp Cloud Volumes Service in die Google Cloud Platform unterstützt wird, identifizieren Sie das Volume anhand seines Volume-Pfads. Der Volumenpfad ist der Teil des Exportpfades des Volumes nach dem `:/`. Wenn der Exportpfad beispielsweise lautet `10.0.0.1:/adroit-jolly-swift`, ist der Volumenpfad `adroit-jolly-swift` .

Beispiel für die Google Cloud Platform

Im folgenden Beispiel wird ein Volume auf dem Backend `gcpcvs_YEppr` mit dem Volume-Pfad von `adroit-jolly-swift` importiert `gcp-cvs`.

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

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

Azure NetApp Dateien

Astra Trident unterstützt den Volume-Import mithilfe des `azure-netapp-files` Treibers.

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

Beispiel: Azure NetApp Files

Das folgende Beispiel importiert ein `azure-netapp-files` Volume auf dem Backend `azurenetappfiles_40517` mit dem Volume-Pfad `importvol1`.

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

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

Passen Sie Volume-Namen und -Beschriftungen an

Mit Astra Trident können Sie erstellten Volumes 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:

1. Volume-Erstellung, -Import und -Klonen
2. Im Fall des ontap-nas-Economy-Treibers entspricht nur der Name des Qtree-Volumes der Namensvorlage.
3. Im Fall des ontap-san-Economy-Treibers entspricht nur der LUN-Name der Namensvorlage.

Einschränkungen

1. Anpassbare Volume-Namen sind nur mit ONTAP On-Premises-Treibern kompatibel.
2. Anpassbare Volume-Namen gelten nicht für vorhandene Volumes.

Wichtige Verhaltensweisen anpassbarer Volumennamen

1. 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.
2. 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.Requ
estName}}"
},
"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

1. 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"}}`.
2. Im Fall des Imports von verwalteten Volumes folgt der Name des Volumes der Namensvorlage, die in der Backend-Definition auf Root-Ebene definiert wurde.
3. Astra Trident unterstützt nicht die Verwendung eines Slice-Operators mit dem Storage-Präfix.
4. Wenn die Vorlagen nicht zu eindeutigen Volume-Namen führen, fügt Astra Trident einige zufällige Zeichen an, um eindeutige Volume-Namen zu erstellen.
5. Wenn der benutzerdefinierte Name für ein NAS-Economy-Volume 64 Zeichen lang ist, benennt Astra 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

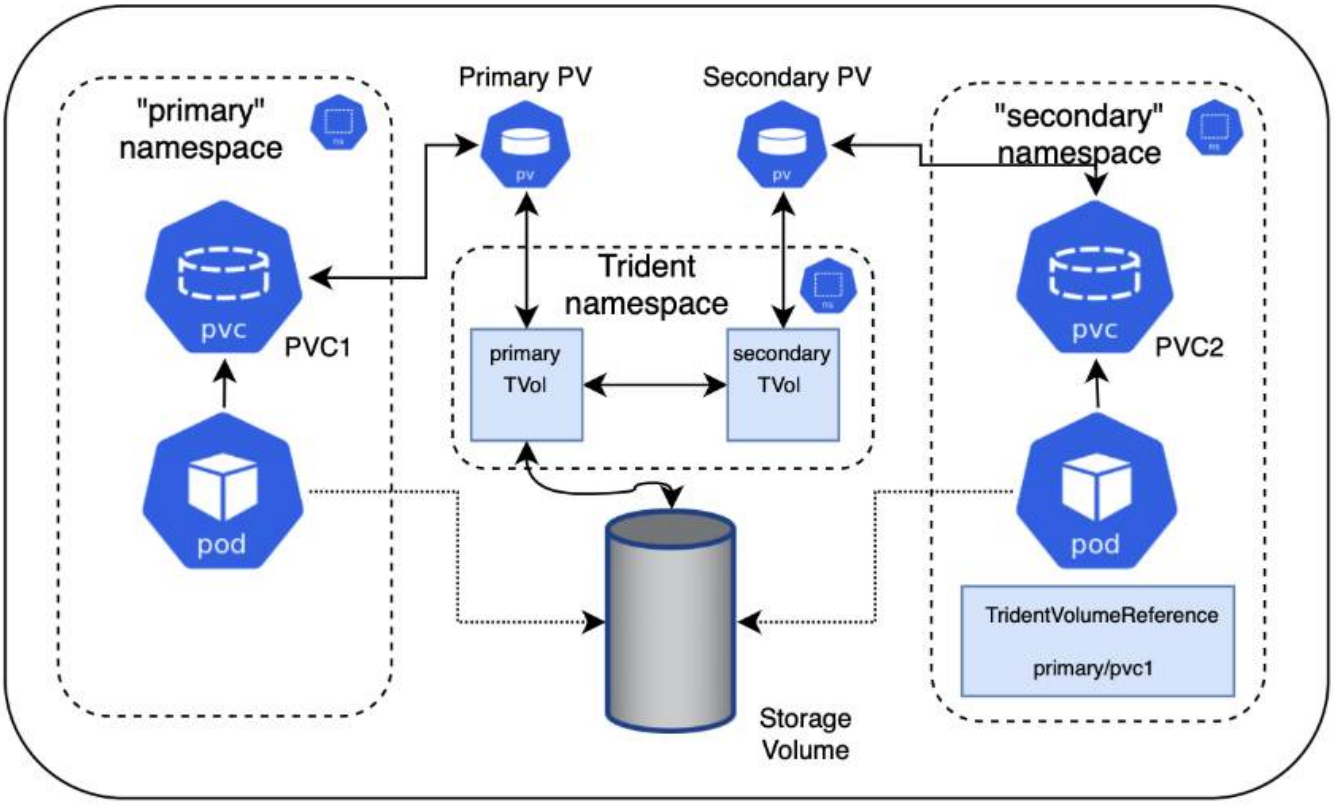
Mit Astra Trident können Sie ein Volume in einem primären Namespace erstellen und es in einem oder mehreren sekundären Namespaces teilen.

Funktionen

Mit dem Astra TridentVolumeReference CR können Sie ReadWriteManche (RWX) NFS-Volumes sicher über einen oder mehrere Kubernetes-Namespaces teilen. 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-Namespaces.



Schnellstart

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

1 Konfigurieren Sie die Quell-PVC für die gemeinsame Nutzung des Volumes

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

2 Erteilen Sie die Berechtigung zum Erstellen eines CR im Zielspeicherort

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

3 Erstellen Sie TridentVolumeReference im Ziel-Namespace

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

4 Erstellen Sie die 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 die PVC (`pvc1`) im Source Namespace, der die Erlaubnis erteilt, mit dem Ziel-Namespace zu teilen (`namespace2`) mit der `shareToNamespace` Annotation.

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

   Astra Trident erstellt das PV und das Back-End NFS Storage Volume.

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

2. **Cluster Admin:** Erstellen Sie die benutzerdefinierte Rolle und kubeconfig, um dem Ziel-Namespace-Eigentümer die Berechtigung zu erteilen, das TridentVolumeReference CR im Ziel-Namespace zu erstellen.
3. **Destination Namespace Owner:** 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`) mit der `shareFromPVC` Anmerkung die Quell-PVC zu bestimmen.

```
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

Astra Trident liest die `shareFromPVC` Annotation auf der Ziel-PVC ein und erstellt das Ziel-PV als 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. Astra Trident entfernt den Zugriff auf das Volume im Quell-Namespace und behält auch andere Namespaces, die das Volume gemeinsam nutzen. Wenn alle Namespaces entfernt werden, die auf dem Volume verweisen, löscht Astra Trident das Volume.

Zum Abfragen untergeordneter Volumes verwenden `tridentctl get`

Mit dem[`tridentctl` Dienstprogramm können Sie den Befehl ausführen `get`, um untergeordnete Volumes zu erhalten. Weitere Informationen finden Sie unter Link:../Trident-reference/tridentctl.html[`tridentctl`

Commands and options].

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

Markierungen:

- `-h, --help`: Hilfe für Bände.
- `--parentOfSubordinate string`: Abfrage auf untergeordneten Quellvolume beschränken.
- `--subordinateOf string`: Abfrage auf Untergebene des Volumens beschränken.

Einschränkungen

- Astra Trident kann nicht verhindern, dass Ziel-Namespaces auf dem Shared 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 Quell-PVC nicht aufheben, indem Sie die Anmerkungen oder `shareFromNamespace` entfernen `shareToNamespace` oder den CR löschen `TridentVolumeReference`. Um den Zugriff zu widerrufen, müssen Sie das untergeordnete PVC löschen.
- Snapshots, Klone 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".

Replizieren Sie Volumes mit SnapMirror

Mit Astra Control Provisioner können Sie Spiegelungsbeziehungen zwischen einem Quell-Volume auf einem Cluster und dem Ziel-Volume auf dem Peering-Cluster erstellen, um Daten für die Disaster Recovery zu replizieren. Sie können eine benutzerdefinierte Ressourcendefinition (CRD, Namested Custom Resource Definition) 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

- **Astra Control Provisioner**: Astra Control Provisioner Version 23.10 oder höher muss sowohl auf den Quell- als auch auf den Ziel-Kubernetes-Clustern vorhanden sein, die ONTAP als Backend verwenden.
- **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" .

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.

- **Astra Control Provisioner und SVM**: Die Peering von Remote-SVMs müssen für die Astra Control Bereitstellung im Ziel-Cluster verfügbar sein.

Unterstützte Treiber

- Die Volume-Replizierung wird von ontap-nas und ontap-san Treibern unterstützt.

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
```

Astra Control Provisioner ruft die internen Informationen für das Volume und den aktuellen DP-Status des Volumes ab und füllt dann das Statusfeld der MirrorRelationship 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"
      ```

Astra Control Provisioner erstellt eine SnapMirror Beziehung zum Namen der konfigurierten Beziehungsrichtlinie (oder dem Standard für ONTAP) und initialisiert sie.

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
```

Astra Control Provisioner überprüft die CRD für die TridentMirrorRelationship und erstellt das Volume nicht, wenn die Beziehung nicht vorhanden ist. Falls die Beziehung besteht, stellt Astra Control Provisioner sicher, dass das neue FlexVol Volume auf eine SVM platziert wird, die mit der in MirrorRelation definierten Remote SVM 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 Astra Control Provisioner über den gewünschten Status informiert. 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

Astra Control Provisioner unterstützt die folgenden Vorgänge für primäre und sekundäre 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 Zielflvxole, 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)

Astra Control Provisioner überprüft nach replizierten PVCs und gibt die Replizierungsbeziehung frei, bevor versucht wird, das Volume zu löschen.

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 Astra Control Provisioner den Löschvorgang abgeschlossen hat. Wenn der für den Löschvorgang ausgewählte TMR bereits den Status *promoted* hat, gibt es keine bestehende Spiegelbeziehung und der TMR wird entfernt und Astra Control Provisioner 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 Astra Control direkt mit dem ONTAP Cluster verbunden ist. 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 *suileded*, *in progress* oder *failed* annehmen.

Astra Control Provisioner Aktivieren

Trident Version 23.10 und höher bieten die Option zur Verwendung von Astra Control Provisioner, damit Benutzer von Astra Control auf erweiterte Funktionen zur Storage-Bereitstellung zugreifen können. Astra Control Provisioner bietet diese erweiterte Funktionalität zusätzlich zu den auf Astra Trident basierenden Standardfunktionen. Mit diesem Verfahren können Sie Astra Control Provisioner aktivieren und installieren.

Im Abonnement für Astra Control Service ist automatisch die Lizenz für die Nutzung von Astra Control Provisioner enthalten.

Bei den neuesten Updates für Astra Control wird Astra Control Provisioner Astra Trident als Storage-bereitstellung und -Orchestrierung ersetzen und für die Verwendung von Astra Control obligatorisch sein. Aus diesem Grund wird dringend empfohlen, Astra Control für die Astra Control-Bereitstellung zu aktivieren. Astra Trident wird weiterhin Open Source bleiben und mit neuen CSI- und anderen Funktionen von NetApp veröffentlicht, gepflegt, unterstützt und auf dem neuesten Stand sein.

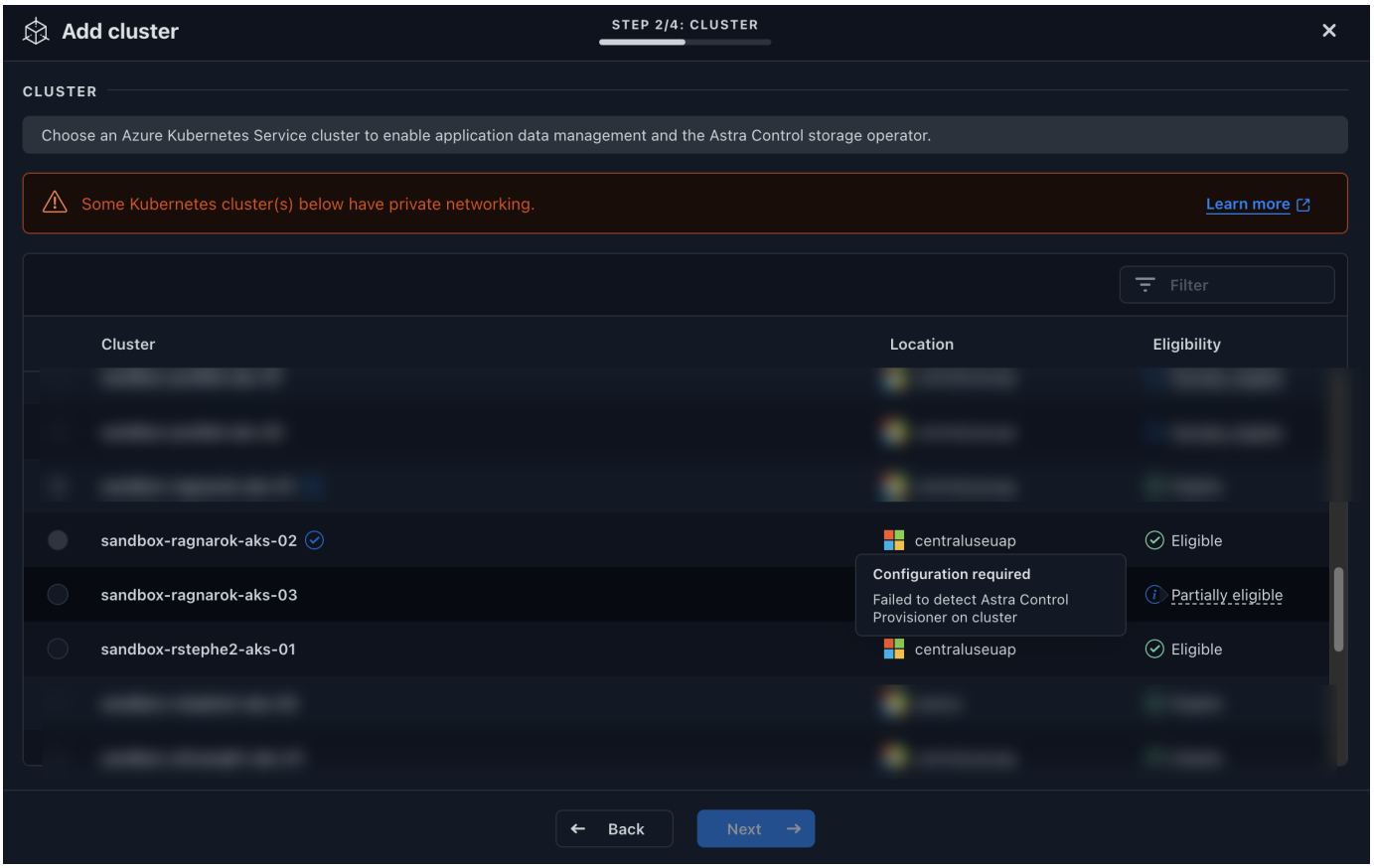
Wie kann ich feststellen, ob ich die Astra Control-Bereitstellung aktivieren muss?

Wenn Sie Astra Control Service einen Cluster hinzufügen, für den Astra Trident zuvor nicht installiert ist, wird der Cluster als markiert `Eligible`. Nach Ihnen "Fügen Sie den Cluster zu Astra Control hinzu"wird der Astra Control Provisioner automatisch aktiviert.

Wenn Ihr Cluster nicht markiert ist `Eligible`, wird er aufgrund einer der folgenden Zeichen markiert `Partially eligible`:

- Es verwendet eine ältere Version von Astra Trident
- In Astra Trident 23.10 wird noch nicht die bereitstellungsoption aktiviert
- Es handelt sich um einen Cluster-Typ, der keine automatische Aktivierung zulässt

In `Partially eligible` Fällen können Sie mit diesen Anweisungen die Astra Control Provisioner für Ihr Cluster manuell aktivieren.



Bevor Sie Astra Control Provisioner aktivieren

Wenn Sie bereits Astra Trident ohne Astra Control Provisioner verwenden und Astra Control Provisioner aktivieren möchten, gehen Sie zuerst wie folgt vor:

- **Wenn Sie Astra Trident installiert haben, bestätigen Sie, dass seine Version innerhalb eines Fensters mit vier Versionen ist**: Sie können ein direktes Upgrade auf Astra Trident 24.02 mit Astra Control Provisioner durchführen, wenn Ihr Astra Trident innerhalb eines Fensters mit vier Versionen von Version 24.02 ist. Sie können beispielsweise direkt von Astra Trident 23.04 auf 24.02 aktualisieren.
- **Bestätigen Sie, dass Ihr Cluster über eine AMD64-Systemarchitektur verfügt**: Das Astra Control Provisioner-Image wird sowohl in AMD64- als auch in ARM64-CPU-Architekturen bereitgestellt, aber nur AMD64 wird von Astra Control unterstützt.

Schritte

1. Rufen Sie die NetApp Astra Control Image-Registry auf:
   a. Melden Sie sich an der Astra Control Service UI an und notieren Sie Ihre Astra Control Konto-ID.
      i. Wählen Sie das Symbol oben rechts auf der Seite.
      ii. Wählen Sie **API-Zugriff**.
      iii. Notieren Sie sich Ihre Konto-ID.
   b. Wählen Sie auf derselben Seite **API-Token generieren** aus und kopieren Sie die API-Token-Zeichenfolge in die Zwischenablage und speichern Sie sie in Ihrem Editor.
   c. Melden Sie sich über Ihre bevorzugte Methode in der Astra Control Registry an:

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

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

2. (Nur benutzerdefinierte Registrierungen) Befolgen Sie diese Schritte, um das Bild in Ihre benutzerdefinierte Registrierung zu verschieben. Wenn Sie keine Registrierung verwenden, befolgen Sie die Schritte des Trident-Operators im Nächster Abschnitt.

> Sie können Podman anstelle von Docker für die folgenden Befehle verwenden. Wenn Sie eine Windows-Umgebung verwenden, wird PowerShell empfohlen.

Docker

a. Rufen Sie das Astra Control Provisioner-Image aus der Registrierung ab:

> Das abgezogene Image unterstützt nicht mehrere Plattformen und unterstützt nur die gleiche Plattform wie der Host, der das Image gezogen hat, wie z. B. Linux AMD64.

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

Beispiel:

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

b. Markieren Sie das Bild:

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

c. Laden Sie das Bild in Ihre benutzerdefinierte Registrierung:

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

Kran

a. Kopieren Sie das Astra Control Provisioner-Manifest in Ihre benutzerdefinierte Registry:

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

3. Stellen Sie fest, ob die ursprüngliche Astra Trident Installationsmethode einen verwendethat.
4. Aktivieren Sie Astra Control Provisioner in Astra Trident mit der ursprünglich verwendeten Installationsmethode:

Astra Trident Betreiber

a. "Laden Sie das Astra Trident Installationsprogramm herunter und extrahieren Sie es".

b. Führen Sie diese Schritte aus, wenn Sie Astra Trident noch nicht installiert haben oder den Operator aus der ursprünglichen Astra Trident-Implementierung entfernt haben:

i. Erstellen des CRD:

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

ii. Erstellen Sie den dreigesichtigen Namespace (`kubectl create namespace trident`) oder bestätigen Sie, dass der dreigesichtigen Namespace noch existiert (`kubectl get all -n trident`). Wenn der Namespace entfernt wurde, erstellen Sie ihn erneut.

c. Update von Astra Trident auf 24.06.0:

> Verwenden Sie für Cluster mit Kubernetes 1.24 oder früher `bundle_pre_1_25.yaml`: Verwenden Sie für Cluster mit Kubernetes 1.25 oder höher `bundle_post_1_25.yaml`.

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

d. Überprüfen Sie, ob Astra Trident ausgeführt wird:

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

Antwort:

```
NAME      AGE
trident   21m
```

e. Wenn Sie eine Registry mit Geheimnissen haben, erstellen Sie ein Geheimnis, mit dem Sie das Astra Control Provisioner-Bild abrufen können:

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

f. Bearbeiten Sie den TridentOrchestrator CR, und nehmen Sie die folgenden Änderungen vor:

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

i. Legen Sie einen benutzerdefinierten Registrierungsort für das Astra Trident-Image fest oder ziehen Sie es aus der Astra Control-Registry (`tridentImage: <my_custom_registry>/trident:24.02.0` oder `tridentImage: netapp/trident:24.06.0`).

ii. Aktivieren Sie Astra Control Provisioner (`enableACP: true`).

iii. Legen Sie den benutzerdefinierten Registrierungsport für das Astra Control Provisioner-Image fest oder ziehen Sie es aus der Astra Control Registry (`acpImage: <my_custom_registry>/trident-acp:24.02.0` oder `acpImage: cr.astra.netapp.io/astra/trident-acp:24.02.0`).

iv. Wenn Sie in diesem Verfahren bereits einmal eingerichtet Geheimnisse der Bildausziehung haben, können Sie diese hier einstellen (`imagePullSecrets: - <secret_name>`). Verwenden Sie den gleichen geheimen Namen, den Sie in den vorherigen Schritten festgelegt haben.

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

g. Speichern und beenden Sie die Datei. Der Bereitstellungsprozess wird automatisch gestartet.

h. Überprüfen Sie, ob der Operator, die Bereitstellung und Replikasets erstellt wurden.

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

Es sollte nur eine Instanz* des Operators in einem Kubernetes-Cluster geben. Erstellen Sie nicht mehrere Implementierungen des Astra Trident Operators.

i. Überprüfen Sie, ob der `trident-acp` Container läuft und der `acpVersion` Status lautet `24.02.0 Installed`:

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

Antwort:

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

Tridentctl

a. "Laden Sie das Astra Trident Installationsprogramm herunter und extrahieren Sie es".

b. "Wenn Sie bereits Astra Trident verwenden, deinstallieren Sie ihn aus dem Cluster, das ihn hostet".

c. Astra Trident mit aktivierter Astra Control Provisioner installieren (`--enable-acp=true`):

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

d. Aktivieren Sie die Astra Control Provisioner-Funktion:

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

Antwort:

```
+----------------+---------------+-------------+ | SERVER
VERSION | CLIENT VERSION | ACP VERSION | +----------------
+----------------+-------------+ | 24.02.0 | 24.02.0 | 24.02.0. |
+----------------+---------------+-------------+
```

Helm

a. Bei Installation von Astra Trident 23.07.1 oder einer früheren Version sind der Operator und andere Komponenten installiert "Deinstallieren" .

b. Wenn auf dem Kubernetes-Cluster 1.24 oder eine frühere Version ausgeführt wird, löschen Sie psp:

```
kubectl delete psp tridentoperatorpod
```

c. Fügen Sie das Helm Repository von Astra Trident hinzu:

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

d. Aktualisieren Sie das Helm-Diagramm:

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

Antwort:

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

e. Auflisten der Bilder:

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

Antwort:

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

f. Stellen Sie sicher, dass Dreizack-Bediener 24.06.0 verfügbar ist:

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

Antwort:

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

g. Verwenden Sie `helm install` eine der folgenden Optionen, die diese Einstellungen enthalten, und führen Sie sie aus:

- Ein Name für Ihren Bereitstellungsort
- Die Version Astra Trident
- Der Name des Bildes für die Astra Control-Bereitstellung
- Das Flag, mit dem die provisionierung aktiviert wird
- (Optional) Ein lokaler Registrierungspfad. Wenn Sie eine lokale Registrierung verwenden, kann sich Ihr "Trident Images" in einer Registrierung oder in verschiedenen Registrierungen befinden, aber alle CSI-Images müssen sich in derselben Registrierung befinden.
- Der Trident Namespace

Optionen

◦ Bilder ohne Registrierung

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

◦ Bilder in einer oder mehreren Registern

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

Mit können `helm list` Sie Installationsdetails wie Name, Namespace, Diagramm, Status, App-Version, und Revisionsnummer.

Falls Sie Probleme bei der Implementierung von Trident mit Helm haben, führen Sie diesen Befehl aus, um Astra Trident vollständig zu deinstallieren:

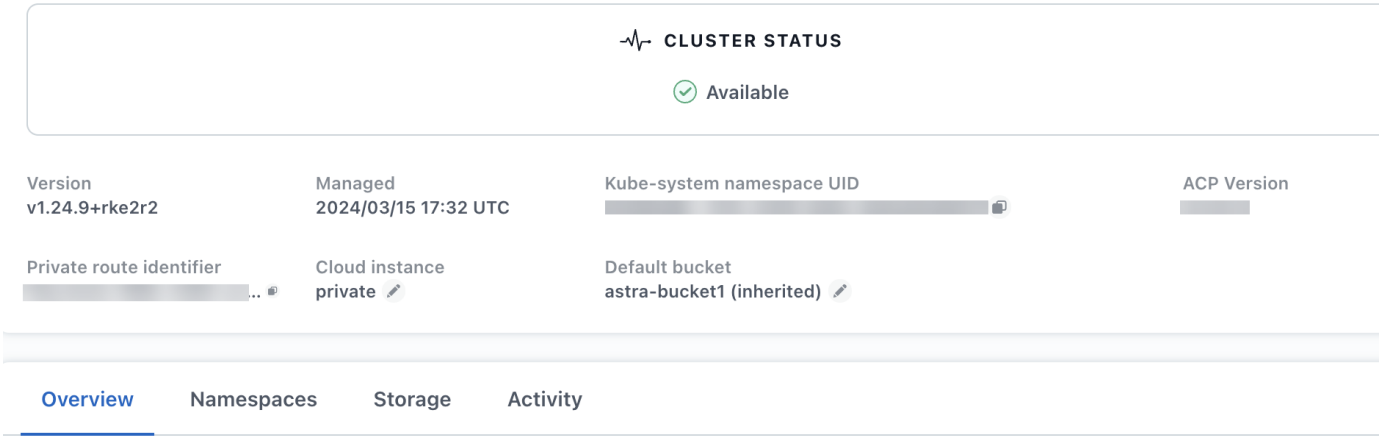
```
./tridentctl uninstall -n trident
```

Nicht "Astra Trident CRDs vollständig entfernen" als Teil Ihrer Deinstallation, bevor Sie versuchen, Astra Control Provisioner wieder zu aktivieren.

Ergebnis

Die Bereitstellungsfunktion von Astra Control ist aktiviert und Sie können alle Funktionen der verwendeten Version verwenden.

Nach der Installation von Astra Control provisioner wird im Cluster, das die provisionierung in der Astra Control UI hostet, ein Feld statt und die aktuelle installierte Versionsnummer angezeigt `ACP version` `Trident version`.



Finden Sie weitere Informationen

- "Dokumentation für Astra Trident Upgrades"

Verwenden Sie die CSI-Topologie

Astra Trident kann selektiv Volumes erstellen und an die in einem Kubernetes-Cluster vorhandenen Nodes 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. Astra Trident verwendet CSI Topology, um die Provisionierung von Volumes für Workloads in einer Multi-Zone-Architektur zu vereinfachen.

Erfahren Sie mehr über die Funktion „CSI-Topologie "Hier"".

Kubernetes bietet zwei unterschiedliche Modi für die Volume-Bindung:

- Mit `VolumeBindingMode` Set to `Immediate` erstellt Astra Trident das Volume ohne jegliche Topologieorientierung. 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 der `VolumeBindingMode` Einstellung auf `WaitForFirstConsumer` wird die Erstellung und Bindung eines persistenten Volumes für eine PVC 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.

> Für den `WaitForFirstConsumer` 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:

- Ein Kubernetes Cluster mit einem "Unterstützte Kubernetes-Version"

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

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

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

Schritt 1: Erstellen Sie ein Topologieorientiertes Backend

Astra Trident Storage-Back-Ends können für die selektive Bereitstellung von Volumes basierend auf Verfügbarkeitszonen ausgelegt werden. 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 einen Teil der Regionen und Zonen enthalten, die in einem Backend bereitgestellt werden, erstellt Astra Trident ein Volume im Backend.

Sie können auch pro Speicherpool definieren `supportedTopologies`. 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 stehen die `region` Etiketten und `zone` für den Speicherort des Speicherpools. `topology.kubernetes.io/region` Und `topology.kubernetes.io/zone` legen Sie fest, wo die Speicherpools genutzt 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:
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
- key: topology.kubernetes.io/zone
  values:
  - us-east1-a
  - us-east1-b
- key: topology.kubernetes.io/region
  values:
  - us-east1
parameters:
  fsType: "ext4"
```

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 das folgende Beispiel `spec`:

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san
spec:
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 weist Kubernetes an, den Pod auf Nodes zu planen, die in der Region vorhanden sind `us-east1`, und aus jedem Node, der in der Zone oder `us-east1-b` vorhanden ist, auszuwählen `us-east1-a`.

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
```

Back-Ends aktualisieren, um sie einzuschließen `supportedTopologies`

Bereits vorhandene Back-Ends können aktualisiert werden, um eine Liste der Verwendung `tridentctl backend update` aufzunehmen `supportedTopologies`. 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 Astra Trident erstellten Volumes erstellen, einen außerhalb von Astra Trident erstellten Snapshot importieren, ein neues Volume aus einem vorhandenen Snapshot erstellen und Volume-Daten aus Snapshots wiederherstellen.

Überblick

Volumen-Snapshot wird unterstützt von `ontap-nas`, , `ontap-nas-flexgroup` `ontap-san` `ontap-san-economy` `solidfire-san` , , , `gcp-cvs`und `azure-netapp-files` 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 Ihre Kubernetes-Distribution den Snapshot Controller und CRDs nicht enthält, finden Sie weitere Informationen unter 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 eine `VolumeSnapshotClass`. Weitere Informationen finden Sie unter "VolumeSnapshotKlasse".
   - Der `driver` verweist auf den Astra Trident CSI-Treiber.
   - `deletionPolicy` Kann oder `Retain` sein `Delete`. Wenn auf festgelegt `Retain`, wird der zugrunde liegende physische Snapshot auf dem Speicher-Cluster auch dann beibehalten, wenn das `VolumeSnapshot` Objekt gelöscht wird.

     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`, und der Name des Snapshots wird auf festgelegt `pvc1-snap`. Ein VolumeSnapshot ist analog zu einer PVC und einem Objekt zugeordnet `VolumeSnapshotContent`, 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 das Objekt für den `pvc1-snap` VolumeSnapshot identifizieren `VolumeSnapshotContent`, indem Sie es beschreiben. Das `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:
    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

Sie können verwenden `dataSource`, um eine PVC mit einem VolumeSnapshot zu erstellen, der als Datenquelle benannt `<pvc-name>` ist. 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 als Datenquelle erstellt `pvc1-snap`.

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

Importieren Sie einen Volume-Snapshot

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

Bevor Sie beginnen

Astra 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 Astra Trident gestartet.
   - Geben Sie den Namen des Back-End-Snapshots in `annotations` als `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`an.
   - Geben Sie `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` in an `snapshotHandle`. Dies ist die einzige Information, die Astra Trident vom externen Snapshotter im Aufruf zur Verfügung gestellt `ListSnapshots` wird.

     Der `<volumeSnapshotContentName>` kann aufgrund von Einschränkungen bei der CR-Benennung nicht immer mit dem Namen des Back-End-Snapshots übereinstimmen.

     Beispiel

     Im folgenden Beispiel wird ein Objekt erstellt `VolumeSnapshotContent`, das auf einen 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>
```

2. **Cluster admin:** Erstellen Sie den `VolumeSnapshot` CR, der das Objekt referenziert `VolumeSnapshotContent`. Damit wird der Zugriff auf die Verwendung des in einem bestimmten Namespace benötigt `VolumeSnapshot`.

   Beispiel

   Im folgenden Beispiel wird ein CR mit dem `import-snap` Namen erstellt `VolumeSnapshot`, der auf den Namen `import-snap-content` verweist `VolumeSnapshotContent`.

```
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. Astra Trident erstellt die `TridentSnapshot`.
   - Der externe Schnapper setzt den `VolumeSnapshotContent` auf `readyToUse` und den `VolumeSnapshot` auf `true`.
   - Trident kehrt zurück `readyToUse=true`.
4. **Jeder Benutzer:** Erstellen Sie ein `PersistentVolumeClaim`, um auf den neuenzu verweisen `VolumeSnapshot`, wobei der `spec.dataSource` (oder `spec.dataSourceRef`) Name der Name ist `VolumeSnapshot`.

   Beispiel

Im folgenden Beispiel wird eine PVC erstellt, die auf den Namen `import-snap` verweist `VolumeSnapshot`.

```
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 der mit den Treibern und `ontap-nas-economy` bereitgestellten Volumes zu ermöglichen `ontap-nas`. Aktivieren Sie das `.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.

Das Snapshot-Verzeichnis ist standardmäßig ausgeblendet, um die maximale Kompatibilität der mit den Treibern und `ontap-nas-economy` bereitgestellten Volumes zu ermöglichen `ontap-nas`. Aktivieren Sie das `.snapshot` Verzeichnis, um Daten von Snapshots direkt wiederherzustellen.

> 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

Astra Control Provisioner ermöglicht mithilfe des (TASR) CR 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.

Astra Control Provisioner unterstützt Snapshot-Wiederherstellung auf dem `ontap-san`, , `ontap-san-economy` `ontap-nas` `ontap-nas-flexgroup` `azure-netapp-files`, , , `gcp-cvs` `solidfire-san` und 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`.

```
cat tasr-pvc1-snapshot.yaml

apiVersion: v1
kind: TridentActionSnapshotRestore
metadata:
  name: this-doesnt-matter
  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/this-doesnt-matter
created
```

Ergebnisse

Mit Astra Control Provisioner werden die Daten aus dem Snapshot wiederhergestellt. Sie können den Status der Snapshot-Wiederherstellung überprüfen.

```
kubectl get tasr -o yaml

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

- In den meisten Fällen versucht die Astra Control Provisioner bei einem Ausfall nicht automatisch einen weiteren Vorgang auszuführen. 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

Wenn Sie ein persistentes Volume mit zugeordneten Snapshots löschen, wird das entsprechende Trident-Volume in einen „Löschzustand“ aktualisiert. Entfernen Sie die Volume Snapshots, um das Astra 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.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. 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 ggf. `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` Ihren Namespace und aktualisieren Sie `namespace` ihn.

Weiterführende Links

- "Volume Snapshots"
- "VolumeSnapshotKlasse"