Provisionierung und Management von Volumes

Astra Trident

NetApp
April 03, 2024

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

A "*PersistentVolume*" (PV) ist eine physische Speicherressource, die vom Clusteradministrator auf einem Kubernetes-Cluster bereitgestellt wird. Der "*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 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: 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 die 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
```

Siehe "Kubernetes und Trident Objekte" Erfahren Sie, wie Storage-Klassen mit dem interagieren `PersistentVolumeClaim` Und Parameter für die Steuerung, wie Astra Trident Volumes provisioniert.

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 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
```

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 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 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 die Volume-Erweiterung für auf bereitgestellte NFS PVS `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `gcp-cvs`, und `azure-netapp-files` Back-Ends:

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
```

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 den Satz `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: 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 als Kubernetes PV 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.

- Aber `StorageClass` Muss auf einer PVC angegeben werden, Astra Trident verwendet diesen Parameter während des Imports 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.
- Astra Trident verwaltet standardmäßig die PVC und benennt die FlexVol und die LUN auf dem Backend um. Sie können die passieren `--no-manage` Flag zum Importieren eines nicht verwalteten Volumes. Wenn Sie verwenden `--no-manage`, Astra Trident führt keine zusätzlichen Operationen auf der PVC oder PV für den Lebenszyklus der Objekte. 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

Verwenden Sie können `tridentctl import` Um ein Volume zu importieren.

Schritte

1. Erstellen der PVC-Datei (Persistent Volume Claim) (beispielsweise `pvc.yaml`), die verwendet werden, um die PVC zu erstellen. Die PVC-Datei sollte enthalten `name`, `namespace`, `accessModes`, und `storageClassName`. Optional können Sie angeben `unixPermissions` In Ihrer PVC-Definition.

   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 die `tridentctl import` Befehl zur Angabe des Namens des Astra Trident Back-End, das das Volume enthält, sowie des Namens, der das Volume auf dem Storage eindeutig identifiziert (z. B. ONTAP FlexVol, Element Volume, Cloud Volumes Service-Pfad). Der `-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 von `ontap-nas` Und `ontap-nas-flexgroup` Treiber.

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

Jedes Volume wurde mit erstellt `ontap-nas` Treiber ist ein FlexVol auf dem ONTAP Cluster. Importieren von FlexVols mit dem `ontap-nas` Der Treiber funktioniert genauso. Eine FlexVol, die bereits auf einem ONTAP Cluster vorhanden ist, kann als importiert werden `ontap-nas` PVC: Ebenso können FlexGroup Volumes importiert werden als `ontap-nas-flexgroup` VES.

Beispiele für ONTAP NAS

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

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` Argument, Astra Trident benennt 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 SAN

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

Astra Trident kann ONTAP SAN FlexVols importieren, die eine einzige LUN enthalten. Dies entspricht dem `ontap-san` Treiber, der für jede PVC und eine LUN innerhalb 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 den um `pvc-<uuid>` Formatieren Sie und die LUN innerhalb der FlexVol bis `lun0`.

Im folgenden Beispiel wird der importiert `ontap-san-managed` FlexVol, die auf dem vorhanden ist `ontap_san_default` Back-End:

```
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

Astra Trident unterstützt die NetApp Element Software und den NetApp HCI Volume-Import über die `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

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

Google Cloud Platform

Astra Trident unterstützt den Volume-Import mithilfe von `gcp-cvs` Treiber.

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 Volume-Pfad ist der Teil des Exportpfades des Volumes nach dem `:/`. Beispiel: Wenn der Exportpfad lautet `10.0.0.1:/adroit-jolly-swift`, Der Volume-Pfad ist `adroit-jolly-swift`.

Beispiel für die Google Cloud Platform

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

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

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

Azure NetApp Dateien

Astra Trident unterstützt den Volume-Import mithilfe von `azure-netapp-files` Treiber.

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`.

Beispiel: Azure NetApp Files

Im folgenden Beispiel wird ein importiert `azure-netapp-files` Volume am 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    |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Ein NFS-Volume kann über Namespaces hinweg genutzt werden

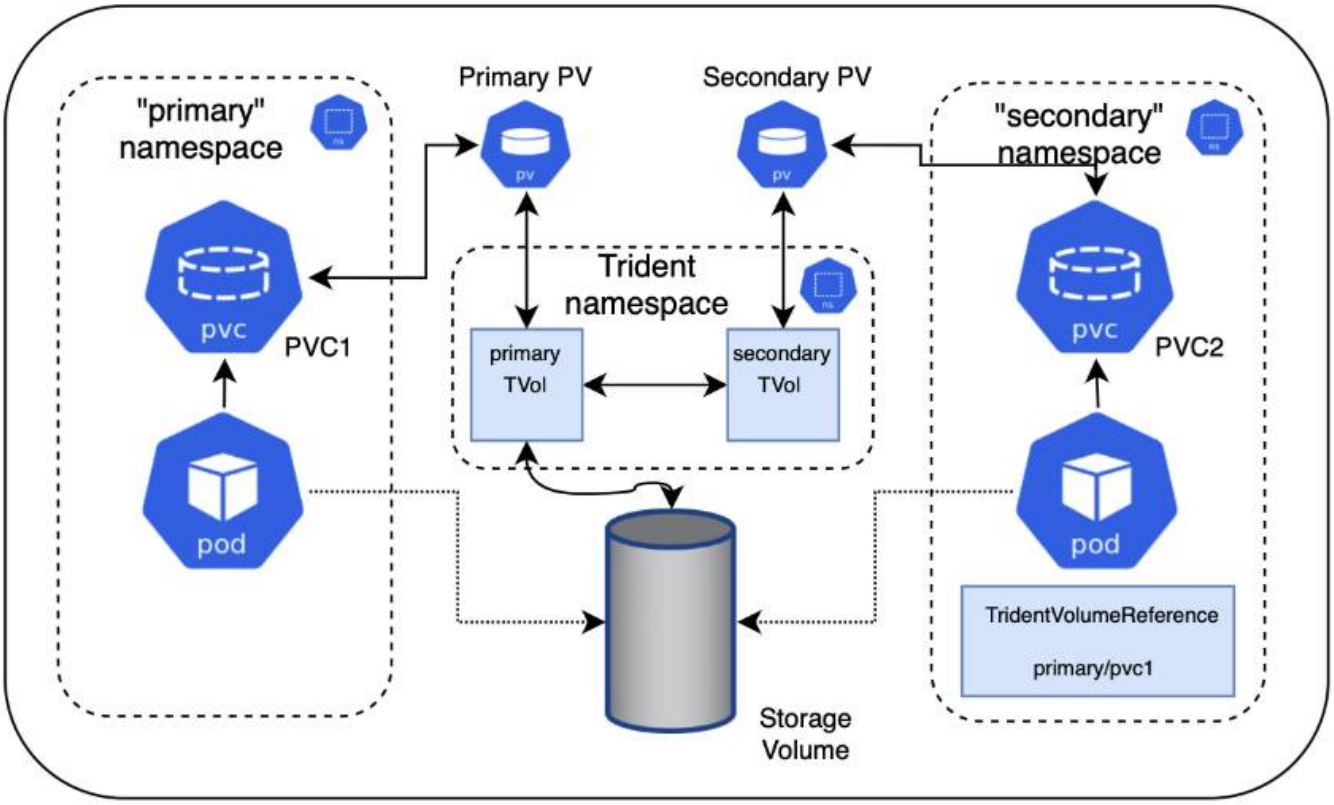
Mit Astra Trident können Sie ein Volume in einem primären Namespace erstellen und es in einem oder mehreren sekundären Namespaces teilen.

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 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
```

   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. 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. **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. **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

Astra Trident liest den `shareFromPVC` Anmerkung auf dem Ziel-PVC und erstellt das Ziel-PV als untergeordnetes Volumen ohne eigene Speicherressource, die auf das Quell-PV verweist und die PV-Quellressource teilt. 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.

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-

Referenz/tridentctl.html[`tridentctl` Befehle und Optionen].

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

Markierungen:

- `-h, --help`: Hilfe für Volumen.
- `--parentOfSubordinate string`: Abfrage auf untergeordnetes Quellvolumen begrenzen.
- `--subordinateOf string`: Abfrage auf Untergebene 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 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, 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".
- Demo ansehen am "NetAppTV".

Verwenden Sie die CSI-Topologie

Astra Trident kann Volumes selektiv erstellen und zu Nodes in einem Kubernetes Cluster verbinden, indem der verwendet wird "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 Topology "Hier".

Kubernetes bietet zwei unterschiedliche Modi für die Volume-Bindung:

- Mit `VolumeBindingMode` Auf einstellen `Immediate`, Astra Trident erstellt das Volume ohne Topologiebewusstsein. Die Volume-Bindung und die dynamische Bereitstellung werden bei der Erstellung des PVC behandelt. Dies ist die Standardeinstellung `VolumeBindingMode` Und ist für Cluster geeignet, die keine Topologiebeschränkungen mehr durchsetzen. Persistente Volumes werden erstellt, ohne von den Planungsanforderungen des anfragenden Pods abhängig zu sein.
- Mit `VolumeBindingMode` Auf einstellen `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:"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 eine Topologiebewusstsein einführen (`topology.kubernetes.io/region` Und `topology.kubernetes.io/zone`). 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 optional mittragen `supportedTopologies` Block, der eine Liste der zu unterstützenden 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 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
  region: Iowa-DC
  zone: Iowa-DC-A
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
- labels:
    workload: dev
  region: Iowa-DC
  zone: Iowa-DC-B
  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:
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 Zonen und die Region bereit, die verwendet werden sollen. Der `netapp-san-us-east1` StorageClass erstellt VES auf dem `san-backend-us-east1` Back-End oben definiert.

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:
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 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 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

Volume Snapshot wird von unterstützt `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, und `azure-netapp-files` Treiber.

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 Astra 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:
    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: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Importieren Sie einen Volume-Snapshot

Astra Trident unterstützt das "Vorab bereitgestellter Snapshot-Prozess von Kubernetes" Damit der Clusteradministrator einen erstellen kann `VolumeSnapshotContent` Objekt- und Import von Snapshots, 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 eine `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 an `annotations` Als `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
   - Angeben `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` In `snapshotHandle`. Dies ist die einzige Information, die Astra Trident vom externen Snapshot in zur Verfügung gestellt wird `ListSnapshots` Anruf.

     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>
```

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 Snapshotter erkennt das neu erstellte `VolumeSnapshotContent` Und führt das aus `ListSnapshots` Anruf. Astra 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.

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

Weiterführende Links

- "Volume Snapshots"
- "VolumeSnapshotKlasse"