



# Durchführung von Volume-Vorgängen

## Astra Trident

NetApp  
April 16, 2024

# Inhalt

- Durchführung von Volume-Vorgängen ..... 1
  - Verwenden Sie die CSI-Topologie ..... 1
  - Arbeiten Sie mit Snapshots ..... 8
  - Erweitern Sie Volumes ..... 12
  - Volumes importieren ..... 19

# Durchführung von Volume-Vorgängen

Erfahren Sie mehr über die Funktionen von Astra Trident zum Management Ihrer Volumes.

- ["Verwenden Sie die CSI-Topologie"](#)
- ["Arbeiten Sie mit Snapshots"](#)
- ["Erweitern Sie Volumes"](#)
- ["Volumes importieren"](#)

## 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"](#). 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 dass sie von den Planungsanforderungen des anfragenden Pods abhängig sind.
- 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:"1e11e4a2108024935ecfcb2912226cedeaafd99df",
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:"1e11e4a2108024935ecfcb2912226cedeaafd99df",
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","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/os":"linux","node-role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/os":"linux","node-role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/os":"linux","node-role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

## Schritt 1: Erstellen Sie ein Topologieorientiertes Backend

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-centrall
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-centrall
  topology.kubernetes.io/zone: us-centrall-a
- topology.kubernetes.io/region: us-centrall
  topology.kubernetes.io/zone: us-centrall-b
storage:
- labels:
    workload: production
    region: Iowa-DC
    zone: Iowa-DC-A
    supportedTopologies:
    - topology.kubernetes.io/region: us-centrall
      topology.kubernetes.io/zone: us-centrall-a
- labels:
    workload: dev
    region: Iowa-DC
    zone: Iowa-DC-B
    supportedTopologies:
    - topology.kubernetes.io/region: us-centrall
      topology.kubernetes.io/zone: us-centrall-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:

```

kubect1 create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubect1 get pvc
NAME          STATUS      VOLUME      CAPACITY      ACCESS MODES      STORAGECLASS
AGE
pvc-san      Pending
2s
kubect1 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
  ----      -
  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: voll
        persistentVolumeClaim:
          claimName: pvc-san
    containers:
      - name: sec-ctx-demo
        image: busybox
        command: [ "sh", "-c", "sleep 1h" ]
        volumeMounts:
          - name: voll
            mountPath: /data/demo
        securityContext:
          allowPrivilegeEscalation: false

```

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 VolumeSnapshots (Volume Snapshot) von persistenten Volumes (PVS) können erstellt werden, um zeitpunktgenaue Kopien von Astra Trident Volumes zu erhalten. Darüber hinaus können Sie aus einem vorhandenen Volume-Snapshot ein neues Volume, auch als „Clone“ bekannt, erstellen. Volume Snapshot wird von `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, und `azure-netapp-files` Treiber.

### Bevor Sie beginnen

Sie müssen über einen externen Snapshot-Controller und benutzerdefinierte Ressourcen-Definitionen (CRDs) verfügen. 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 [Bereitstellung eines Volume Snapshot Controllers](#).



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

## Schritt 1: Erstellen Sie ein VolumeSnapshotClass

Dieses Beispiel erstellt eine Volume-Snapshot-Klasse.

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

Der driver Zeigt auf den CSI-Treiber von Astra Trident. `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.

Weitere Informationen finden Sie unter Link: [../trident-reference/Objects.HTML#kubernetes-volumesnapshotclass-Objects\[VolumeSnapshotClass\]](https://trident-reference/Objects.HTML#kubernetes-volumesnapshotclass-Objects[VolumeSnapshotClass]).

## Schritt 2: Erstellen Sie einen Schnappschuss eines vorhandenen PVC

In diesem Beispiel wird ein Snapshot eines vorhandenen PVC erstellt.

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

In diesem Beispiel wird der Snapshot für ein PVC mit dem Namen erstellt `pvcl` Der Name des Snapshots lautet `pvcl-snap`.

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

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

Dadurch wurde ein erstellt `VolumeSnapshot` Objekt: Ein `VolumeSnapshot` ist analog zu einem PVC und einem zugeordnet `VolumeSnapshotContent` Objekt, das den tatsächlichen Snapshot darstellt.

Es ist möglich, die zu identifizieren `VolumeSnapshotContent` Objekt für das `pvcl-snap` `VolumeSnapshot` wird beschrieben.

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

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.

### Schritt 3: PVCs aus VolumeSnapshots erstellen

Das Beispiel erstellt ein PVC anhand eines Snapshots:

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

`dataSource` Zeigt an, dass das PVC mit dem Namen `VolumeSnapshot` erstellt werden muss `pvcl-snap` Als Quelle der Daten. Damit beauftragt Astra Trident, aus dem Snapshot ein PVC zu erstellen. Nachdem die PVC erstellt wurde, kann sie an einem Pod befestigt und wie jedes andere PVC verwendet werden.



Wenn Sie ein persistentes Volume mit zugeordneten Snapshots löschen, wird das entsprechende Trident-Volume in einen „Löschzustand“ aktualisiert. Damit das Astra Trident Volume gelöscht werden kann, sollten die Snapshots des Volume entfernt werden.

## Bereitstellung eines Volume Snapshot Controllers

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

### Schritte

1. Erstellen von Volume Snapshot-CRDs.

```
cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

- Erstellen Sie den Snapshot-Controller im gewünschten Namespace. Bearbeiten Sie die YAML-Manifeste unten, um den Namespace zu ändern.

```
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
```

## Weiterführende Links

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

## 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-Volumen

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



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

### Überblick

Die Erweiterung eines iSCSI-PV umfasst die folgenden Schritte:

- Bearbeiten der StorageClass-Definition zum Festlegen des `allowVolumeExpansion` Feld an `true`.

- Bearbeiten der PVC-Definition und Aktualisieren der `spec.resources.requests.storage` Um die neu gewünschte Größe zu reflektieren, die größer als die ursprüngliche Größe sein muss.
- Das Anbringen des PV muss an einen Pod angehängt werden, damit die Größe geändert werden kann. 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.

Das folgende Beispiel zeigt, wie die Erweiterung von iSCSI PVS funktioniert.

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

```
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

```
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

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:

```

kubect1 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
kubect1 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-Volumen

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 `kubect1 edit storageclass` bearbeiten, 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 GB:

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

#### Schritt 4: Validieren Sie die Erweiterung

Sie können die korrekte Größenänderung validieren, indem Sie die Größe des PVC, des PV und des Astra Trident Volume überprüfen:

```

kubect1 get pvc ontapnas20mb
NAME                STATUS      VOLUME
CAPACITY           ACCESS MODES  STORAGECLASS  AGE
ontapnas20mb      Bound      pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  1Gi
RWO                ontapnas          4m44s

kubect1 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`.

### Treiber, die den Volumenimport unterstützen

In dieser Tabelle sind die Treiber aufgeführt, die den Import von Volumes unterstützen, und die Version, in der sie eingeführt wurden.

Treiber	Freigabe
ontap-nas	19.04
ontap-nas-flexgroup	19.04
solidfire-san	19.04
azure-netapp-files	19.04

Treiber	Freigabe
gcp-cvs	19.04
ontap-san	19.04

## Warum sollte ich Volumes importieren?

Es gibt verschiedene Anwendungsfälle für den Import eines Volumes in Trident:

- Eine Anwendung erreichen und ihren vorhandenen Datensatz erneut verwenden
- Verwenden eines Klons eines Datensatzes für eine kurzlebige Anwendung
- Neuerstellung eines fehlerhaften Kubernetes-Clusters
- Migration von Applikationsdaten während der Disaster Recovery

## Wie funktioniert der Import?

Die PVC-Datei (Persistent Volume Claim) wird vom Importprozess des Volumes zur Erstellung des PVC verwendet. Die PVC-Datei sollte mindestens die Felder Name, Namespace, accessModes und storageClassName enthalten, wie im folgenden Beispiel dargestellt.

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

Der `tridentctl` Client wird verwendet, um ein vorhandenes Storage Volume zu importieren. Trident importiert das Volume, indem Volume-Metadaten gespeichert und die PVC und das PV erstellt werden.

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

Zum Importieren eines Storage-Volumes geben Sie den Namen des Astra Trident Backends mit dem Volume sowie den Namen an, der das Volume auf dem Storage eindeutig identifiziert (z. B. ONTAP FlexVol, Element Volume, CVS Volume Path). Das Storage-Volume muss Lese-/Schreibzugriff ermöglichen und über das angegebene Astra Trident-Back-End zugänglich sein. Der `-f` String Argument ist erforderlich und gibt den Pfad zur YAML- oder JSON-PVC-Datei an.

Erhält Astra Trident die Anfrage für das Importvolumen, wird die vorhandene Volume-Größe festgelegt und im PVC 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.

Wenn ein Volume mit dem importiert wird `--no-manage` Argument: Trident führt für den Lebenszyklus der Objekte keine zusätzlichen Operationen an der PVC oder PV durch. Da Trident PV- und PVC-Ereignisse für ignoriert `--no-manage` Objekte, das Speichervolumen wird nicht gelöscht, wenn das PV gelöscht wird. Andere Vorgänge, wie z. B. der Volume-Klon und die Volume-Größe, werden ebenfalls ignoriert. 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.

Trident 19.07 und höher verarbeiten den Anhang von PVS und mountet das Volume im Rahmen des Imports. Bei Importen mit früheren Versionen von Astra Trident gibt es keine Vorgänge im Datenpfad. Der Volume-Import überprüft nicht, ob das Volume gemountet werden kann. Wenn beim Import des Volumes ein Fehler gemacht wird (beispielsweise ist `StorageClass` falsch), können Sie die Zurückgewinnungsrichtlinie für das PV in wiederherstellen `retain`, Löschen der PVC und PV, und Wiederversuchen des Volumenimportbefehls.

## ontap-nas **Und** ontap-nas-flexgroup **Importe**

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.



Ein ONTAP Volume muss vom Typ `rw` aufweisen, um von Trident zu importieren. Wenn ein Volume vom Typ `dp` verwendet wird, es ein SnapMirror Ziel-Volume ist. Sie sollten die gespiegelte Beziehung unterbrechen, bevor Sie das Volume in Trident importieren.



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

Zum Beispiel, um ein Volume mit dem Namen zu importieren `managed_volume` Auf einem Backend mit dem Namen `ontap_nas`, Verwenden Sie den folgenden Befehl:

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

So importieren Sie ein Volume mit dem Namen `unmanaged_volume` (Auf dem `ontap_nas` backend), die Trident nicht verwaltet, verwenden Sie den folgenden Befehl:

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

Bei Verwendung des `--no-manage` Argument: Trident umbenannt oder validiert nicht, ob das Volume angehängt war. Der Volumenimport schlägt fehl, wenn das Volume nicht manuell gemountet wurde.



Ein zuvor vorhandener Fehler beim Importieren von Volumes mit benutzerdefinierten UnixPermissions wurde behoben. Sie können `unixPermissions` in Ihrer PVC-Definition oder Back-End-Konfiguration angeben und Astra Trident anweisen, das Volume entsprechend zu importieren.

## ontap-san Importieren

Astra Trident kann auch ONTAP SAN FlexVols importieren, die eine einzelne LUN enthalten. Dies entspricht dem `ontap-san` Treiber, der für jede PVC und eine LUN innerhalb der FlexVol eine FlexVol erstellt. Sie können das verwenden `tridentctl import` Befehl in gleicher Weise wie in anderen Fällen:

- Geben Sie den Namen des an `ontap-san` Back-End:
- Geben Sie den Namen der zu importierenden FlexVol an. Beachten Sie, dass diese FlexVol nur eine LUN enthält, die importiert werden muss.
- Geben Sie den Pfad der PVC-Definition an, die mit dem verwendet werden muss `-f` Flagge.
- Wählen Sie zwischen PVC-Verwaltung oder -Management. Standardmäßig verwaltet Trident die PVC und benennt die FlexVol und LUN auf dem Back-End um. Um als nicht verwaltetes Volume zu importieren, übergeben Sie den `--no-manage` Flagge.



Beim Importieren eines nicht verwalteten `ontap-san` Volume, Sie sollten sicherstellen, dass die LUN in der FlexVol benannt ist `lun0` Und ist einer Initiatorgruppe mit den gewünschten Initiatoren zugeordnet. Astra Trident übernimmt dies automatisch für einen verwalteten Import.

Astra Trident importiert dann den FlexVol und verknüpft ihn mit der PVC-Definition. Astra Trident ist auch für die FlexVol bekannt `pvc-<uuid>` Formatieren Sie und die LUN innerhalb der FlexVol bis `lun0`.



Es wird empfohlen, Volumes zu importieren, die keine aktiven Verbindungen haben. Wenn Sie ein aktiv verwendetes Volume importieren möchten, klonen Sie zuerst das Volume und führen Sie dann den Import durch.

### Beispiel

Um den zu importieren `ontap-san-managed FlexVol`, die auf dem vorhanden ist `ontap_san_default` Back-End, führen Sie das aus `tridentctl import` Befehl als:

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



Ein ONTAP-Volume muss vom Typ `rw` sein, um von Astra Trident importiert werden zu können. Wenn ein Volume vom Typ `dp` ist, ist es ein SnapMirror Ziel-Volume. Sie sollten die Spiegelbeziehung brechen, bevor Sie das Volume in Astra Trident importieren.

### element Importieren

Mit Trident können Sie NetApp Element Software/NetApp HCI Volumes in Ihr Kubernetes Cluster importieren. Sie brauchen den Namen Ihres Astra Trident Backend, und den eindeutigen Namen des Volumes und der PVC-Datei als Argumente für die `tridentctl import` Befehl.

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

```



Der Elementtreiber unterstützt doppelte Volume-Namen. Wenn es doppelte Volume-Namen gibt, gibt Trident Volume Import Prozess einen Fehler zurück. Als Workaround können Sie das Volume klonen und einen eindeutigen Volume-Namen bereitstellen. Importieren Sie dann das geklonte Volume.

## gcp-cvs Importieren



Für den Import eines durch die NetApp Cloud Volumes Service in GCP gesicherten Volumes sollten Sie das Volume nach seinem Volume-Pfad anstelle seines Namens identifizieren.

Um einen zu importieren gcp-cvs Datenträger auf dem Back-End aufgerufen gcpcvs\_YEppr Mit dem Volume-Pfad von adroit-jolly-swift, Verwenden Sie den folgenden Befehl:

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

```



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

## azure-netapp-files Importieren

Um einen zu importieren azure-netapp-files Datenträger auf dem Back-End aufgerufen `azurenetaappfiles_40517` Mit dem Volume-Pfad `importvoll1`, Ausführen des folgenden Befehls:

```
tridentctl import volume azurenetaappfiles_40517 importvoll1 -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        |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```



Der Volume-Pfad für das ANF-Volumen ist im Mount-Pfad nach dem `:/` vorhanden. Beispiel:  
Wenn der Mount-Pfad lautet `10.0.0.2:/importvoll1`, Der Volume-Pfad ist `importvoll1`.

## Copyright-Informationen

Copyright © 2024 NetApp. Alle Rechte vorbehalten. Gedruckt in den USA. Dieses urheberrechtlich geschützte Dokument darf ohne die vorherige schriftliche Genehmigung des Urheberrechtinhabers in keiner Form und durch keine Mittel – weder grafische noch elektronische oder mechanische, einschließlich Fotokopieren, Aufnehmen oder Speichern in einem elektronischen Abrufsystem – auch nicht in Teilen, vervielfältigt werden.

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

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

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

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

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

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

## Markeninformationen

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