# Referenz

## Trident

NetApp
September 26, 2025

# Inhalt

# Referenz

## Trident-Ports

Erfahren Sie mehr über die Ports, die Trident für die Kommunikation verwendet.

### Trident-Ports

Trident kommuniziert über folgende Ports:

| Port | Zweck |
|---|---|
| 8443 | Backchannel HTTPS |
| 8001 | Endpunkt der Prometheus Kennzahlen |
| 8000 | Trident REST-Server |
| 17546 | Anschluss für Liveness/Readiness-Sonde, der von Trident Demonset-Pods verwendet wird |

Der Anschluss für die Liveness/Readiness-Sonde kann während der Installation mit dem Flag geändert `--probe-port` werden. Es ist wichtig, sicherzustellen, dass dieser Port nicht von einem anderen Prozess auf den Worker-Knoten verwendet wird.

## Trident REST-API

Sie "Tridentctl-Befehle und -Optionen" sind die einfachste Möglichkeit zur Interaktion mit der Trident REST-API, können Sie, falls gewünscht, den REST-Endpunkt direkt verwenden.

### Wann die REST-API verwendet werden soll

DIE REST-API ist nützlich für erweiterte Installationen, die Trident als Standalone-Binärdatei in Implementierungen ohne Kubernetes verwenden.

Zur Verbesserung der Sicherheit ist das Trident `REST API` standardmäßig auf localhost beschränkt, wenn es innerhalb eines Pods ausgeführt wird. Um dieses Verhalten zu ändern, müssen Sie das Argument von Trident in der POD-Konfiguration festlegen `-address`.

### REST-API wird verwendet

Für Beispiele, wie diese APIs aufgerufen werden, übergeben Sie das (`-d`Flag debug ). Weitere Informationen finden Sie unter "Managen Sie Trident mit tridentctl".

Die API funktioniert wie folgt:

#### GET

**GET <trident-address>/trident/v1/<object-type>**

Listet alle Objekte dieses Typs auf.

**GET <trident-address>/trident/v1/<object-type>/<object-name>**

Ruft die Details des benannten Objekts ab.

#### POST

**POST <trident-address>/trident/v1/<object-type>**

Erstellt ein Objekt des angegebenen Typs.

- Eine JSON-Konfiguration für das zu erstellenden Objekt erforderlich. Informationen zur Spezifikation der einzelnen Objekttypen finden Sie unter "Managen Sie Trident mit tridentctl".
- Falls das Objekt bereits vorhanden ist, variiert das Verhalten: Back-Ends aktualisiert das vorhandene Objekt, während alle anderen Objekttypen den Vorgang nicht ausführen.

#### Löschen

**DELETE <trident-address>/trident/v1/<object-type>/<object-name>**

Löscht die benannte Ressource.

> Es existieren weiterhin Volumes, die mit Back-Ends oder Storage-Klassen verbunden sind. Diese müssen separat gelöscht werden. Weitere Informationen finden Sie unter "Managen Sie Trident mit tridentctl".

# Befehlszeilenoptionen

Trident bietet mehrere Befehlszeilenoptionen für den Trident Orchestrator. Sie können diese Optionen verwenden, um Ihre Bereitstellung zu ändern.

## Protokollierung

**-debug**

Aktiviert die Debugging-Ausgabe.

**-loglevel <level>**

Legt die Protokollierungsebene fest (Debug, Info, Warn, ERROR, Fatal). Standardmäßig Info.

## Kubernetes

**-k8s_pod**

Verwenden Sie diese Option oder `-k8s_api_server`, um den Kubernetes-Support zu aktivieren. Durch diese Einstellung verwendet Trident die Zugangsdaten für das Kubernetes-Servicekonto eines Pods, um den API-Server zu kontaktieren. Dies funktioniert nur, wenn Trident als Pod in einem Kubernetes-Cluster mit aktivierten Service-Konten ausgeführt wird.

**-k8s_api_server <insecure-address:insecure-port>**

Verwenden Sie diese Option oder `-k8s_pod`, um den Kubernetes-Support zu aktivieren. Bei Angabe von stellt Trident über die angegebene unsichere Adresse und den angegebenen Port eine Verbindung zum

Kubernetes-API-Server her. Dadurch kann Trident außerhalb eines Pods bereitgestellt werden. Es werden jedoch nur unsichere Verbindungen zum API-Server unterstützt. Um eine sichere Verbindung herzustellen, implementieren Sie Trident in einem Pod mit der `-k8s_pod` Option.

### Docker

**`-volume_driver <name>`**

Treibername, der bei der Registrierung des Docker-Plug-ins verwendet wird. Standardmäßig ist `netapp` .

**`-driver_port <port-number>`**

Hören Sie auf diesen Port statt auf einen UNIX-Domain-Socket.

**`-config <file>`**

Erforderlich; Sie müssen diesen Pfad zu einer Back-End-Konfigurationsdatei angeben.

### RUHE

**`-address <ip-or-host>`**

Gibt die Adresse an, auf der der REST-Server von Trident hören soll. Standardmäßig localhost. Wenn auf dem localhost zuhören und in einem Kubernetes Pod ausgeführt werden, ist der ZUGRIFF auf DIE REST-Schnittstelle nicht direkt von außerhalb des Pods möglich. Verwenden Sie `-address ""`, um den Zugriff auf die REST-Schnittstelle über die POD-IP-Adresse zu ermöglichen.

Die Trident REST-Schnittstelle kann nur für die Wiedergabe unter 127.0.0.1 (für IPv4) oder [::1] (für IPv6) konfiguriert werden.

**`-port <port-number>`**

Gibt den Port an, auf dem der REST-Server von Trident lauschen soll. Die Standardeinstellung ist 8000.

**`-rest`**

Aktiviert die REST-Schnittstelle. Standardmäßig auf „true“ gesetzt.

# Kubernetes und Trident Objekte

Kubernetes und Trident lassen sich über REST-APIs miteinander interagieren, indem Objekte gelesen und geschrieben werden. Es gibt verschiedene Ressourcenobjekte, die die Beziehung zwischen Kubernetes und Trident, Trident und Storage sowie Kubernetes und Storage vorschreiben. Einige dieser Objekte werden über Kubernetes verwaltet, andere wiederum über Trident.

## Wie interagieren die Objekte miteinander?

Am einfachsten ist es, die Objekte, deren Bedeutung und ihre Interaktion zu verstehen, wenn ein Kubernetes-Benutzer eine einzelne Storage-Anfrage bearbeitet:

1. Ein Benutzer erstellt ein, das `PersistentVolumeClaim` eine neue Anforderung einer bestimmten Größe von einem Kubernetes `StorageClass` anfordert `PersistentVolume`, das zuvor vom Administrator konfiguriert wurde.

2. Kubernetes `StorageClass` identifiziert Trident als bereitstellung und enthält Parameter, die Trident sagen, wie ein Volume für die angeforderte Klasse bereitgestellt werden kann.
3. Trident betrachtet seinen eigenen `StorageClass` Namen mit dem gleichen Namen, der die Abgleichung identifiziert `Backends` und `StoragePools` den es zur Bereitstellung von Volumes für die Klasse verwenden kann.
4. Trident stellt Storage auf einem passenden Back-End bereit und erstellt zwei Objekte: Ein `PersistentVolume` in Kubernetes, das Kubernetes über die Suche, das Mounten und die Behandlung des Volume informiert, und ein Volume in Trident, das die Beziehung zwischen dem und dem tatsächlichen Storage beibehält. `PersistentVolume`
5. Kubernetes bindet das `PersistentVolumeClaim` an das neue `PersistentVolume`. Pods, die das Mount von PersistentVolume auf jedem Host enthalten `PersistentVolumeClaim`, auf dem es ausgeführt wird.
6. Ein Benutzer erstellt `VolumeSnapshot` mithilfe eines s, das auf Trident verweist, eine einer vorhandenen PVC `VolumeSnapshotClass`.
7. Trident identifiziert das dem PVC zugeordnete Volume und erstellt einen Snapshot des Volumes auf dem Back-End. Es erstellt auch ein `VolumeSnapshotContent`, das Kubernetes über die Identifizierung des Snapshots anweist.
8. Ein Benutzer kann ein Verwenden `VolumeSnapshot` als Quelle erstellen `PersistentVolumeClaim`.
9. Trident identifiziert den erforderlichen Snapshot und führt die gleichen Schritte aus, die beim Erstellen von A und A `Volume` erforderlich sind `PersistentVolume`.

Für weitere Informationen zu Kubernetes-Objekten empfehlen wir, den Abschnitt der Kubernetes-Dokumentation zu lesen "Persistente Volumes".

## `PersistentVolumeClaim`Kubernetes Objekte

Ein Kubernetes- `PersistentVolumeClaim`Objekt ist eine Anforderung von Storage, der von einem Kubernetes-Cluster-Benutzer erstellt wird.

Zusätzlich zur Standardspezifikation können Benutzer mit Trident die folgenden Volume-spezifischen Anmerkungen angeben, wenn sie die in der Back-End-Konfiguration festgelegten Standardeinstellungen überschreiben möchten:

| Anmerkung | Volume-Option | Unterstützte Treiber |
|---|---|---|
| trident.netapp.io/fileSystem | Dateisystem | ontap-san, solidfire-san, ontap-san-Economy |
| trident.netapp.io/cloneFromPVC | KlonSourceVolume | ontap-nas, ontap-san, solidfire-san, Azure-netapp-Dateien, gcp-cvs, ontap-san-Ökonomie |
| trident.netapp.io/splitOnClone | SPlitOnClone | ontap-nas, ontap-san |
| trident.netapp.io/protocol | Protokoll | Alle |
| trident.netapp.io/exportPolicy | Exportpolitik | ontap-nas, ontap-nas-Economy, ontap-nas-Flexgroup |
| trident.netapp.io/snapshotPolicy | SnapshotPolicy | ontap-nas, ontap-nas-Economy, ontap-nas-Flexgroup, ontap-san |

| Anmerkung | Volume-Option | Unterstützte Treiber |
|---|---|---|
| trident.netapp.io/snapshotReserve | SnapshotReserve | ontap-nas, ontap-nas-Flexgroup, ontap-san, gcp-cvs |
| trident.netapp.io/snapshotDirectory | SnapshotDirectory | ontap-nas, ontap-nas-Economy, ontap-nas-Flexgroup |
| trident.netapp.io/unixPermissions | UnxPermissions | ontap-nas, ontap-nas-Economy, ontap-nas-Flexgroup |
| trident.netapp.io/blockSize | Blocksize | solidfire-san |

Wenn das erstellte PV über die Zurückgewinnungsrichtlinie verfügt `Delete`, löscht Trident sowohl das PV als auch das Back-Volume, wenn das PV freigegeben wird (d. h. wenn der Benutzer die PVC löscht). Sollte die Löschaktion fehlschlagen, markiert Trident den PV als solche und wiederholt den Vorgang periodisch, bis er erfolgreich ist oder der PV manuell gelöscht wird. Wenn das PV die Richtlinie verwendet `Retain`, ignoriert Trident sie und geht davon aus, dass der Administrator sie von Kubernetes und dem Back-End bereinigt, sodass das Volume vor dem Entfernen gesichert oder inspiziert werden kann. Beachten Sie, dass das Löschen des PV nicht dazu führt, dass Trident das Backing-Volume löscht. Sie sollten es mit der REST API entfernen (`tridentctl`).

Trident unterstützt die Erstellung von Volume Snapshots anhand der CSI-Spezifikation: Sie können einen Volume Snapshot erstellen und ihn als Datenquelle zum Klonen vorhandener PVCs verwenden. So können zeitpunktgenaue Kopien von PVS in Form von Snapshots Kubernetes zugänglich gemacht werden. Die Snapshots können dann verwendet werden, um neue PVS zu erstellen. Schauen Sie sich an `On-Demand Volume Snapshots`, um zu sehen, wie das funktionieren würde.

Trident liefert außerdem die `cloneFromPVC` Annotationen und `splitOnClone` zum Erstellen von Klonen. Mit diesen Anmerkungen können Sie eine PVC klonen, ohne die CSI-Implementierung verwenden zu müssen.

Hier ist ein Beispiel: Wenn ein Benutzer bereits eine PVC aufgerufen hat `mysql`, kann der Benutzer eine neue PVC erstellen, die über die Anmerkung aufgerufen wird `mysqlclone`, wie `trident.netapp.io/cloneFromPVC: mysql`z.B. . Mit diesem Anmerkungsset klont Trident das Volume, das dem mysql PVC entspricht, anstatt ein Volume von Grund auf neu bereitzustellen.

Berücksichtigen Sie folgende Punkte:

- Wir empfehlen das Klonen eines inaktiven Volumes.
- Ein PVC und sein Klon sollten sich im gleichen Kubernetes Namespace befinden und dieselbe Storage-Klasse haben.
- Bei den `ontap-nas` und `ontap-san`-Treibern könnte es wünschenswert sein, die PVC-Beschriftung in Verbindung mit `trident.netapp.io/cloneFromPVC` einzustellen `trident.netapp.io/splitOnClone`. Mit `trident.netapp.io/splitOnClone` Set-auf teilt Trident das geklonte Volume vom übergeordneten Volume auf `true` und entkoppelt damit den Lebenszyklus des geklonten Volume vollständig von seinem übergeordneten Volume, was den Verlust einiger Storage-Effizienz bedeutet. Wenn Sie diese Einstellung nicht festlegen oder auf `false` diese Einstellung setzen `trident.netapp.io/splitOnClone`, verringert sich der Speicherplatzverbrauch im Backend auf Kosten des Erstellens von Abhängigkeiten zwischen den übergeordneten und den Klon-Volumes, sodass das übergeordnete Volume nicht gelöscht werden kann, es sei denn, der Klon wird zuerst gelöscht. Ein Szenario, in dem das Aufteilen des Klons sinnvoll ist, ist das Klonen eines leeren Datenbank-Volumes, in dem erwartet wird, dass das Volume und der zugehörige Klon eine große Divergenz sind. Es profitieren nicht von der Storage-Effizienz des ONTAP.

Das `sample-input` Verzeichnis enthält Beispiele für PVC-Definitionen für die Verwendung mit Trident. Eine vollständige Beschreibung der Parameter und Einstellungen zu Trident Volumes finden Sie unter.

## `PersistentVolume`Kubernetes Objekte

Ein Kubernetes- `PersistentVolume`Objekt ist ein Storage-Element, der dem Kubernetes Cluster zur Verfügung gestellt wird. Es weist einen Lebenszyklus auf, der unabhängig vom POD ist, der ihn nutzt.

Trident erstellt `PersistentVolume` auf Basis der bereitstehenden Volumes automatisch Objekte und registriert sie beim Kubernetes-Cluster. Sie sollten diese nicht selbst verwalten.

Wenn Sie eine PVC erstellen, die sich auf ein Trident-basiertes bezieht `StorageClass`, stellt Trident ein neues Volume mit der entsprechenden Speicherklasse bereit und registriert ein neues PV für dieses Volume. Bei der Konfiguration des bereitgestellten Volume und des entsprechenden PV befolgt Trident folgende Regeln:

- Trident generiert einen PV-Namen für Kubernetes mit einem internen Namen, der zur Bereitstellung des Storage verwendet wird. In beiden Fällen wird sichergestellt, dass die Namen in ihrem Geltungsbereich eindeutig sind.
- Die Größe des Volumens entspricht der gewünschten Größe in der PVC so genau wie möglich, obwohl es möglicherweise auf die nächste zuteilbare Menge aufgerundet werden, je nach Plattform.

## `StorageClass`Kubernetes Objekte

Kubernetes- `StorageClass`Objekte werden mithilfe des Namens in angegeben `PersistentVolumeClaims`, um Storage mit einem Satz von Eigenschaften bereitzustellen. Die Storage-Klasse selbst gibt die zu verwendenden bereitstellungsunternehmen an und definiert die Eigenschaftengruppe in Bezug auf die provisionierung von.

Es handelt sich um eines von zwei grundlegenden Objekten, die vom Administrator erstellt und verwaltet werden müssen. Das andere ist das Trident Back-End-Objekt.

Ein Kubernetes- `StorageClass`Objekt, das Trident verwendet, sieht folgendermaßen aus:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters:
  <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Diese Parameter sind Trident-spezifisch und Trident erläutert die Bereitstellung von Volumes für die Klasse.

Parameter der Storage-Klasse sind:

| Attribut | Typ | Erforderlich | Beschreibung |
|---|---|---|---|
| Merkmale | Zuordnen einer Zeichenfolge[string] | Nein | Weitere Informationen finden Sie im Abschnitt Attribute unten |
| Storage Pools | Zuordnen[String]StringList | Nein | Zuordnung von Back-End-Namen zu Listen von Storage-Pools innerhalb |
| Zusätzlich StoragePools | Zuordnen[String]StringList | Nein | Zuordnung von Back-End-Namen zu Listen von Storage-Pools innerhalb |
| Unter Ausnahme von StoragePools | Zuordnen[String]StringList | Nein | Zuordnung von Back-End-Namen zu Listen von Storage-Pools innerhalb |

Storage-Attribute und ihre möglichen Werte können in Auswahlebene und Kubernetes-Attribute des Storage-Pools klassifiziert werden.

### Auswahlebene für Storage-Pools

Diese Parameter bestimmen, welche in Trident gemanagten Storage Pools zur Bereitstellung von Volumes eines bestimmten Typs verwendet werden sollten.

| Attribut | Typ | Werte | Angebot | Anfrage | Unterstützt von |
|---|---|---|---|---|---|
| Medien[1] | Zeichenfolge | hdd, Hybrid, ssd | Pool enthält Medien dieser Art. Beides bedeutet Hybrid | Medientyp angegeben | ontap-nas, ontap-nas-Economy, ontap-nas-Flexgroup, ontap-san, solidfire-san |
| Bereitstellungstyp | Zeichenfolge | Dünn, dick | Pool unterstützt diese Bereitstellungsmethode | Bereitstellungsmethode angegeben | Thick: All ONTAP; Thin: Alle ONTAP und solidfire-san |
| BackendType | Zeichenfolge | ontap-nas, ontap-nas-Economy, ontap-nas-Flexgroup, ontap-san, solidfire-san, gcp-cvs, Azure-netapp-Files, ontap-san-Wirtschaftlichkeit | Pool gehört zu dieser Art von Backend | Back-End angegeben | Alle Treiber |
| Snapshots | bool | Richtig, falsch | Pool unterstützt Volumes mit Snapshots | Volume mit aktivierten Snapshots | ontap-nas, ontap-san, solidfire-san, gcp-cvs |

| Attribut | Typ | Werte | Angebot | Anfrage | Unterstützt von |
|---|---|---|---|---|---|
| Klone | bool | Richtig, falsch | Pool unterstützt das Klonen von Volumes | Volume mit aktivierten Klonen | ontap-nas, ontap-san, solidfire-san, gcp-cvs |
| Verschlüsselung | bool | Richtig, falsch | Pool unterstützt verschlüsselte Volumes | Volume mit aktivierter Verschlüsselung | ontap-nas, ontap-nas-Economy, ontap-nas-Flexgroups, ontap-san |
| IOPS | Int | Positive Ganzzahl | Pool kann IOPS in diesem Bereich garantieren | Volume hat diese IOPS garantiert | solidfire-san |

[1]: Nicht unterstützt von ONTAP Select-Systemen

In den meisten Fällen beeinflussen die angeforderten Werte direkt die Bereitstellung. Wenn Sie beispielsweise Thick Provisioning anfordern, entsteht ein Volume mit Thick Provisioning. Ein Element Storage-Pool nutzt jedoch den angebotenen IOPS-Minimum und das Maximum, um QoS-Werte anstelle des angeforderten Werts festzulegen. In diesem Fall wird der angeforderte Wert nur verwendet, um den Speicherpool auszuwählen.

Idealerweise können Sie `attributes` allein die Qualitäten des Storage modellieren, den Sie zur Erfüllung der Anforderungen einer bestimmten Klasse benötigen. Trident erkennt und wählt automatisch Speicherpools aus, die mit den von Ihnen angegebenen *allen* übereinstimmen `attributes`.

Wenn Sie nicht in der Lage sind, `attributes` automatisch die richtigen Pools für eine Klasse auszuwählen, können Sie die Parameter und `additionalStoragePools` verwenden `storagePools`, um die Pools weiter zu verfeinern oder sogar eine bestimmte Gruppe von Pools auszuwählen.

Mit dem Parameter können Sie `storagePools` die Anzahl der Pools, die mit den angegebenen übereinstimmen, weiter einschränken `attributes`. Mit anderen Worten: Trident verwendet die Kreuzung von Pools, die durch die Parameter und `storagePools` für das Provisioning identifiziert `attributes` werden. Sie können entweder allein oder beides zusammen verwenden.

Sie können den Parameter verwenden `additionalStoragePools`, um den Pool-Satz zu erweitern, den Trident für das Provisioning verwendet, unabhängig von den durch die Parameter und `storagePools` ausgewählten Pools `attributes`.

Sie können den Parameter verwenden `excludeStoragePools`, um den Satz von Pools zu filtern, den Trident für das Provisioning verwendet. Mit diesem Parameter werden alle Pools entfernt, die übereinstimmen.

In den `storagePools` Parametern und `additionalStoragePools` hat jeder Eintrag das Formular `<backend>:<storagePoolList>`, wobei `<storagePoolList>` eine kommagetrennte Liste von Speicherpools für das angegebene Backend ist. Beispielsweise könnte ein Wert für `additionalStoragePools` wie aussehen `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze` . Diese Listen akzeptieren Regex-Werte sowohl für das Backend als auch für Listenwerte. Sie können verwenden `tridentctl get backend`, um die Liste der Back-Ends und deren Pools zu erhalten.

### Attribute für Kubernetes

Diese Attribute haben keine Auswirkung auf die Auswahl von Storage-Pools/Back-Ends, die von Trident während der dynamischen Provisionierung durchgeführt werden. Stattdessen liefern diese Attribute einfach Parameter, die von Kubernetes Persistent Volumes unterstützt werden. Worker-Knoten sind für die Erstellung von Dateisystem-Operationen verantwortlich und benötigen möglicherweise Dateisystem-Dienstprogramme, wie z. B. xfsprogs.

| Attribut | Typ | Werte | Beschreibung | Wichtige Faktoren | Kubernetes-Version |
|---|---|---|---|---|---|
| Fstype | Zeichenfolge | Ext4, ext3, xfs | Der Filesystem-Typ für Block-Volumes | solidfire-san, ontap-nas, ontap-nas-Economy, ontap-nas-Flexgroup, ontap-san, ontap-san-Ökonomie | Alle |
| VolumeErweiterung | boolesch | Richtig, falsch | Aktivieren oder deaktivieren Sie die Unterstützung für das Vergrößern der PVC-Größe | ontap-nas, ontap-nas-Ökonomie, ontap-nas-Flexgroup, ontap-san, ontap-san-Ökonomie, solidfire-san, gcp-cvs, Azure-netapp-Files | 1.11+ |
| VolumeBindingmodus | Zeichenfolge | Sofort, WaitForFirstConsumer | Legen Sie fest, wann Volume Binding und dynamische Bereitstellung stattfindet | Alle | 1.19 - 1.26 |

- Mit dem `fsType` Parameter wird der gewünschte Dateisystemtyp für SAN-LUNs gesteuert. Außerdem verwendet Kubernetes die Anwesenheit von `fsType` in einer Storage-Klasse, um anzugeben, dass ein Dateisystem vorhanden ist. Die Volume-Eigentumsrechte können nur mit dem Sicherheitskontext eines Pods gesteuert werden `fsGroup`, wenn `fsType` festgelegt ist. Eine Übersicht über die Einstellung der Volume-Eigentumsrechte mithilfe des `fsGroup` Kontexts finden Sie unter"Kubernetes: Einen Sicherheitskontext für einen Pod oder Container konfigurieren". Kubernetes setzt diesen `fsGroup` Wert nur ein, wenn:

  - `fsType` Wird in der Storage-Klasse festgelegt.
  - Der PVC-Zugriffsmodus ist RWO.

  Für NFS-Speichertreiber ist bereits ein Dateisystem als Teil des NFS-Exports vorhanden. Um die Storage-Klasse zu verwenden, `fsGroup` muss noch ein angegeben werden `fsType`. Sie können es auf oder einen Wert ungleich Null setzen `nfs`.

- Weitere Details zur Volume-Erweiterung finden Sie unter"Erweitern Sie Volumes".
- Das Trident Installer-Paket enthält mehrere Beispiele für Speicherklassen-Definitionen für die Verwendung mit Trident in `sample-input/storage-class-*.yaml`. Durch das Löschen einer Kubernetes-Storage-Klasse wird auch die entsprechende Trident-Storage-Klasse gelöscht.

## `VolumeSnapshotClass`Kubernetes Objekte

Kubernetes- `VolumeSnapshotClass`Objekte sind analog zu `StorageClasses`. Sie helfen, mehrere Speicherklassen zu definieren und werden von Volume-Snapshots referenziert, um den Snapshot der erforderlichen Snapshot-Klasse zuzuordnen. Jeder Volume Snapshot ist einer einzelnen Volume-Snapshot-Klasse zugeordnet.

Ein `VolumeSnapshotClass` sollte von einem Administrator definiert werden, um Snapshots zu erstellen. Eine Volume-Snapshot-Klasse wird mit folgender Definition erstellt:

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

Der `driver` gibt an Kubernetes an, dass Anforderungen von Volume-Snapshots der `csi-snapclass` Klasse von Trident verarbeitet werden. Der `deletionPolicy` gibt die Aktion an, die ausgeführt werden soll, wenn ein Snapshot gelöscht werden muss. Wenn `deletionPolicy` auf festgelegt ist `Delete`, werden die Volume-Snapshot-Objekte sowie der zugrunde liegende Snapshot auf dem Speicher-Cluster entfernt, wenn ein Snapshot gelöscht wird. Wenn Sie diese Einstellung auf setzen `Retain`, bedeutet dies, dass `VolumeSnapshotContent` der physische Snapshot beibehalten wird.

## `VolumeSnapshot`Kubernetes Objekte

Ein Kubernetes- `VolumeSnapshot`Objekt ist eine Anforderung zur Erstellung eines Snapshots eines Volumes. So wie eine PVC eine von einem Benutzer erstellte Anfrage für ein Volume darstellt, besteht bei einem

Volume-Snapshot die Anforderung eines Benutzers, einen Snapshot eines vorhandenen PVC zu erstellen.

Wenn eine Volume-Snapshot-Anfrage eingeht, managt Trident automatisch die Erstellung des Snapshots für das Volume auf dem Backend und legt den Snapshot durch Erstellen eines eindeutigen Objekts dar. `VolumeSnapshotContent` Sie können Snapshots aus vorhandenen VES erstellen und die Snapshots als Datenquelle beim Erstellen neuer VES verwenden.

> ⓘ Der Lebenszyklus eines VolumeSnapshots ist unabhängig von der Quelle PVC: Ein Snapshot bleibt auch nach dem Löschen der Quelle PVC erhalten. Beim Löschen eines PVC mit zugehörigen Snapshots markiert Trident das Backing-Volume für dieses PVC in einem **Deleting** -Zustand, entfernt es aber nicht vollständig. Das Volume wird entfernt, wenn alle zugehörigen Snapshots gelöscht werden.

## `VolumeSnapshotContent`Kubernetes Objekte

Ein Kubernetes- `VolumeSnapshotContent`Objekt ist ein Snapshot, der von einem bereits bereitgestellten Volume erstellt wurde. Er ist analog zu einem `PersistentVolume` und bedeutet einen bereitgestellten Snapshot auf dem Storage-Cluster. Wenn ein Snapshot erstellt wird, behält das Objekt, ähnlich wie `PersistentVolumeClaim` Objekte `VolumeSnapshotContent` von und `PersistentVolume`, eine Eins-zu-eins-Zuordnung zu dem `VolumeSnapshot` Objekt bei, das die Snapshot-Erstellung angefordert hatte.

Das `VolumeSnapshotContent` Objekt enthält Details, die den Snapshot eindeutig identifizieren, z. B. `snapshotHandle` . Dies `snapshotHandle` ist eine eindeutige Kombination aus dem Namen des PV und dem Namen des `VolumeSnapshotContent` Objekts.

Wenn eine Snapshot-Anfrage eingeht, erstellt Trident den Snapshot auf dem Back-End. Nachdem der Snapshot erstellt wurde, konfiguriert Trident ein `VolumeSnapshotContent` Objekt und legt den Snapshot der Kubernetes-API vor.

> ⓘ In der Regel müssen Sie das Objekt nicht verwalten `VolumeSnapshotContent`. Eine Ausnahme ist, wenn Sie außerhalb von Trident erstellen möchten"Importieren Sie einen Volume-Snapshot".

## `CustomResourceDefinition`Kubernetes Objekte

Kubernetes Custom Ressourcen sind Endpunkte in der Kubernetes API, die vom Administrator definiert werden und zum Gruppieren ähnlicher Objekte verwendet werden. Kubernetes unterstützt das Erstellen individueller Ressourcen zum Speichern einer Sammlung von Objekten. Sie können diese Ressourcendefinitionen erhalten, indem Sie ausführen `kubectl get crds`.

CRDs (Custom Resource Definitions) und die zugehörigen Objektmetadaten werden durch Kubernetes im Metadatenspeicher gespeichert. Dadurch ist kein separater Speicher für Trident erforderlich.

Trident verwendet `CustomResourceDefinition` Objekte, um die Identität von Trident Objekten wie Trident Back-Ends, Trident Storage-Klassen und Trident Volumes zu erhalten. Diese Objekte werden von Trident gemanagt. Darüber hinaus werden im CSI-Volume-Snapshot-Framework einige CRS-IDs verwendet, die zum Definieren von Volume-Snapshots erforderlich sind.

CRDs stellen ein Kubernetes-Konstrukt dar. Objekte der oben definierten Ressourcen werden von Trident erstellt. Ein einfaches Beispiel: Wenn ein Backend mit erstellt `tridentctl` wird, wird ein entsprechendes `tridentbackends` CRD-Objekt für den Verbrauch durch Kubernetes erstellt.

Beachten Sie die folgenden CRDs von Trident:

- Wenn Trident installiert ist, werden eine Reihe von CRDs erstellt und können wie alle anderen Ressourcentypen verwendet werden.
- Wenn Sie Trident mit dem Befehl deinstallieren `tridentctl uninstall`, werden Trident-Pods gelöscht, die erstellten CRDs werden jedoch nicht bereinigt. Informationen dazu, wie Trident vollständig entfernt und neu konfiguriert werden kann, finden Sie unter"Deinstallieren Sie Trident".

## Trident-Objekte `StorageClass`

Trident erstellt passende Storage-Klassen für Kubernetes- `StorageClass`Objekte, die in ihrem Feld „bereitstellung“ angegeben werden `csi.trident.netapp.io`. Der Name der Storage-Klasse stimmt mit dem Kubernetes-Objekt überein `StorageClass`, das sie darstellt.

> Mit Kubernetes werden diese Objekte automatisch erstellt, wenn ein Kubernetes `StorageClass`, das Trident als bereitstellungsunternehmen verwendet, registriert wird.

Storage-Klassen umfassen eine Reihe von Anforderungen für Volumes. Trident stimmt diese Anforderungen mit den in jedem Storage-Pool vorhandenen Attributen überein. Ist dieser Storage-Pool ein gültiges Ziel für die Bereitstellung von Volumes anhand dieser Storage-Klasse.

Sie können Storage-Klassen-Konfigurationen erstellen, um Storage-Klassen direkt über DIE REST API zu definieren. Bei Kubernetes-Implementierungen erwarten wir jedoch, dass sie bei der Registrierung neuer Kubernetes-Objekte erstellt werden `StorageClass`.

## Trident Back-End-Objekte

Back-Ends stellen die Storage-Anbieter dar, über die Trident Volumes bereitstellt. Eine einzelne Trident Instanz kann eine beliebige Anzahl von Back-Ends managen.

> Dies ist einer der beiden Objekttypen, die Sie selbst erstellen und verwalten. Die andere ist das Kubernetes- `StorageClass`Objekt.

Weitere Informationen zum Erstellen dieser Objekte finden Sie unter "Back-Ends werden konfiguriert".

## Trident-Objekte `StoragePool`

Storage-Pools stellen die verschiedenen Standorte dar, die für die Provisionierung an jedem Back-End verfügbar sind. Für ONTAP entsprechen diese Aggregaten in SVMs. Bei NetApp HCI/SolidFire entsprechen diese den vom Administrator festgelegten QoS-Bands. Für Cloud Volumes Service entsprechen diese Regionen Cloud-Provider. Jeder Storage-Pool verfügt über eine Reihe individueller Storage-Attribute, die seine Performance-Merkmale und Datensicherungsmerkmale definieren.

Im Gegensatz zu den anderen Objekten hier werden Storage-Pool-Kandidaten immer automatisch erkannt und gemanagt.

## Trident-Objekte `Volume`

Volumes sind die grundlegende Bereitstellungseinheit, die Back-End-Endpunkte umfasst, wie NFS-Freigaben und iSCSI-LUNs. In Kubernetes entsprechen diese direkt `PersistentVolumes` . Wenn Sie ein Volume erstellen, stellen Sie sicher, dass es über eine Storage-Klasse verfügt, die bestimmt, wo das Volume

zusammen mit einer Größe bereitgestellt werden kann.

- In Kubernetes werden diese Objekte automatisch gemanagt. Sie können sich anzeigen lassen, welche Bereitstellung von Trident bereitgestellt wurde.
- Wenn Sie ein PV mit den zugehörigen Snapshots löschen, wird das entsprechende Trident-Volume auf den Status **Löschen** aktualisiert. Damit das Trident Volume gelöscht werden kann, sollten Sie die Snapshots des Volume entfernen.

Eine Volume-Konfiguration definiert die Eigenschaften, über die ein bereitgestelltes Volume verfügen sollte.

| Attribut | Typ | Erforderlich | Beschreibung |
|---|---|---|---|
| Version | Zeichenfolge | Nein | Version der Trident API („1“) |
| Name | Zeichenfolge | ja | Name des zu erstellenden Volumes |
| Storage Class | Zeichenfolge | ja | Storage-Klasse, die bei der Bereitstellung des Volumes verwendet werden muss |
| Größe | Zeichenfolge | ja | Größe des Volumes, das in Byte bereitgestellt werden soll |
| Protokoll | Zeichenfolge | Nein | Zu verwendenden Protokolltyp; „Datei“ oder „Block“ |
| InternalName | Zeichenfolge | Nein | Name des Objekts auf dem Storage-System, das von Trident generiert wird |
| KlonSourceVolume | Zeichenfolge | Nein | ONTAP (nas, san) & SolidFire-*: Name des Volumes aus dem geklont werden soll |
| SPlitOnClone | Zeichenfolge | Nein | ONTAP (nas, san): Den Klon von seinem übergeordneten Objekt trennen |
| SnapshotPolicy | Zeichenfolge | Nein | ONTAP-*: Die Snapshot-Richtlinie zu verwenden |
| SnapshotReserve | Zeichenfolge | Nein | ONTAP-*: Prozentsatz des für Schnappschüsse reservierten Volumens |
| Exportpolitik | Zeichenfolge | Nein | ontap-nas*: Richtlinie für den Export zu verwenden |
| SnapshotDirectory | bool | Nein | ontap-nas*: Ob das Snapshot-Verzeichnis sichtbar ist |

| Attribut | Typ | Erforderlich | Beschreibung |
|---|---|---|---|
| UnxPermissions | Zeichenfolge | Nein | ontap-nas*: Anfängliche UNIX-Berechtigungen |
| Blocksize | Zeichenfolge | Nein | SolidFire-*: Block-/Sektorgröße |
| Dateisystem | Zeichenfolge | Nein | Typ des Filesystems |

Trident wird beim Erstellen des Volume generiert `internalName`. Dies besteht aus zwei Schritten. Zuerst wird das Speicherpräfix (entweder der Standard oder das Präfix in der Backend-Konfiguration) dem Volume-Namen vorangestellt `trident`, was zu einem Namen des Formulars führt `<prefix>-<volume-name>`. Anschließend wird der Name desinfiziert und die im Backend nicht zulässigen Zeichen ersetzt. Für ONTAP-Back-Ends ersetzt er Bindestriche durch Unterstriche (der interne Name lautet also `<prefix>_<volume-name>` ). Bei Element-Back-Ends werden Unterstriche durch Bindestriche ersetzt.

Sie können Volume-Konfigurationen verwenden, um Volumes direkt mit der REST-API bereitzustellen, doch in Kubernetes-Implementierungen erwarten wir, dass die meisten Benutzer die standardmäßige Kubernetes-Methode verwenden `PersistentVolumeClaim`. Trident erstellt dieses Volume-Objekt automatisch im Rahmen des Bereitstellungsprozesses.

## Trident-Objekte `Snapshot`

Snapshots sind eine zeitpunktgenaue Kopie von Volumes, die zur Bereitstellung neuer Volumes oder für Restores verwendet werden kann. In Kubernetes entsprechen diese direkt `VolumeSnapshotContent` Objekten. Jeder Snapshot ist einem Volume zugeordnet, das die Quelle der Daten für den Snapshot ist.

Jedes `Snapshot` Objekt enthält die nachfolgend aufgeführten Eigenschaften:

| Attribut | Typ | Erforderlich | Beschreibung |
|---|---|---|---|
| Version | Zeichenfolge | Ja. | Version der Trident API („1“) |
| Name | Zeichenfolge | Ja. | Name des Trident Snapshot-Objekts |
| InternalName | Zeichenfolge | Ja. | Name des Trident Snapshot-Objekts auf dem Storage-System |
| VolumeName | Zeichenfolge | Ja. | Name des Persistent Volume, für das der Snapshot erstellt wird |
| VolumeInternalName | Zeichenfolge | Ja. | Name des zugehörigen Trident-Volume-Objekts auf dem Storage-System |

In Kubernetes werden diese Objekte automatisch gemanagt. Sie können sich anzeigen lassen, welche Bereitstellung von Trident bereitgestellt wurde.

Bei der Erstellung einer Kubernetes- `VolumeSnapshot`Objektanforderung erstellt Trident ein Snapshot-Objekt auf dem zugrunde liegende Storage-System. Die `internalName` des

Snapshot-Objekts wird durch die Kombination des Präfixes mit dem `UID` des `VolumeSnapshot` Objekts generiert `snapshot-` (z. B. `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660` ). `volumeName` Und `volumeInternalName` werden mit den Details des Backing-Volumes gefüllt.

## Trident `ResourceQuota`-Objekt

Die Trident-Deamonset-Technologie nutzt eine `system-node-critical` Prioritätsklasse – die höchste in Kubernetes verfügbare Klasse –, um sicherzustellen, dass Trident Volumes während des ordnungsgemäßen Shutdowns identifizieren und bereinigen kann. Trident-Dämonset-Pods vermeiden Workloads mit einer niedrigeren Priorität in Clustern, bei denen der Ressourcendruck hoch ist.

Um dies zu erreichen, verwendet Trident ein `ResourceQuota` Objekt, um sicherzustellen, dass eine „systemNode-kritische“ Prioritätsklasse auf dem Trident-Dämonenset erfüllt ist. Vor der Bereitstellung und der Erstellung von Dämonensets sucht Trident nach dem `ResourceQuota` Objekt und wendet es an, falls es nicht erkannt wird.

Wenn Sie mehr Kontrolle über die Standardkontingente und Prioritätsklasse benötigen, können Sie ein Objekt mithilfe des Helm-Diagramms erzeugen `custom.yaml` oder konfigurieren `ResourceQuota`.

Im Folgenden finden Sie ein Beispiel für ein `ResourceQuota`Objekt mit Priorität des Trident-Dämonenset.

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
     matchExpressions:
       - operator : In
         scopeName: PriorityClass
         values: ["system-node-critical"]
```

Weitere Informationen zu Ressourcenquoten finden Sie unter "Kubernetes: Ressourcenkontingente".

### Bereinigen Sie sich `ResourceQuota`, wenn die Installation fehlschlägt

In dem seltenen Fall, in dem die Installation nach der Erstellung des Objekts fehlschlägt `ResourceQuota`, versuchen Sie zuerst"Deinstallation", und installieren Sie dann erneut.

Wenn das nicht funktioniert, entfernen Sie das Objekt manuell `ResourceQuota`.

### Entfernen `ResourceQuota`

Wenn Sie die Kontrolle über Ihre eigene Ressourcenzuweisung bevorzugen, können Sie das Trident-Objekt mit dem folgenden Befehl entfernen `ResourceQuota`:

```
kubectl delete quota trident-csi -n trident
```

# Pod Security Standards (PSS) und Security Context Constraints (SCC)

Kubernetes Pod Security Standards (PSS) und Pod Security Policies (PSP) definieren Berechtigungsebenen und schränken das Verhalten von Pods ein. OpenShift Security Context Constraints (SCC) definieren ebenfalls die Pod-Einschränkung speziell für die OpenShift Kubernetes Engine. Um diese Anpassung zu ermöglichen, aktiviert Trident bestimmte Berechtigungen während der Installation. In den folgenden Abschnitten werden die von Trident festgelegten Berechtigungen beschrieben.

PSS ersetzt Pod Security Policies (PSP). PSP war in Kubernetes v1.21 veraltet und wird in v1.25 entfernt. Weitere Informationen finden Sie unter "Kubernetes: Sicherheit".

## Erforderlicher Kubernetes-Sicherheitskontext und zugehörige Felder

| Berechtigung | Beschreibung |
|---|---|
| Privileged | Bei CSI müssen Mount-Punkte bidirektional sein. Das Trident Node-POD muss einen privilegierten Container ausführen. Weitere Informationen finden Sie unter "Kubernetes: Mount-Ausbreitung". |
| Host-Netzwerk | Erforderlich für den iSCSI-Daemon. `iscsiadm` Verwaltet iSCSI-Mounts und verwendet Host-Netzwerke zur Kommunikation mit dem iSCSI-Daemon. |
| Host-IPC | NFS nutzt Prozesskommunikation (IPC) mit dem NFSD. |
| Host-PID | Erforderlich für den Start `rpc-statd` von NFS. Trident fragt Hostprozesse ab, um festzustellen, ob `rpc-statd` vor dem Mounten von NFS-Volumes ausgeführt wird. |
| Sorgen | Diese `SYS_ADMIN` Funktion ist Bestandteil der Standardfunktionen für privilegierte Container. Docker setzt beispielsweise diese Funktionen für privilegierte Container:<br>`CapPrm: 0000003fffffffff`<br>`CapEff: 0000003fffffffff` |
| Abt | Seccomp-Profil ist in privilegierten Containern immer „unbeschränkt“; daher kann es in Trident nicht aktiviert werden. |

| Berechtigung | Beschreibung |
| --- | --- |
| SELinux | Auf OpenShift werden privilegierte Container in der Domäne („Super Privileged Container“) ausgeführt `spc_t`, und nicht privilegierte Container werden in der Domäne ausgeführt `container_t`. Auf `containerd`, bei `container-selinux` installed werden alle Container in der Domain ausgeführt `spc_t`, was SELinux effektiv deaktiviert. Aus diesem Grund wird Trident den Containern nicht hinzugefügt `seLinuxOptions`. |
| DAC | Privilegierte Container müssen als Root ausgeführt werden. Nicht privilegierte Container werden als Root ausgeführt, um auf unix-Sockets zuzugreifen, die von CSI benötigt werden. |

## Pod-Sicherheitsstandards (PSS)

| Etikett | Beschreibung | Standard |
| --- | --- | --- |
| `pod-security.kubernetes.io/enforce` `pod-security.kubernetes.io/enforce-version` | Ermöglicht die Aufnahme der Trident Controller und Knoten im Namespace für die Installation. Ändern Sie nicht die Namespace-Bezeichnung. | `enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>` |

Das Ändern der Namespace-Labels kann dazu führen, dass Pods nicht geplant werden, ein „Error Creating: …“ oder „Warnung: trident-csi-…“. Überprüfen Sie in diesem Fall, ob die Namespace-Bezeichnung für `privileged` geändert wurde. Falls ja, installieren Sie Trident neu.

## Pod-Sicherheitsrichtlinien (PSP)

| Feld | Beschreibung | Standard |
| --- | --- | --- |
| `allowPrivilegeEscalation` | Privilegierte Container müssen die Eskalation von Berechtigungen ermöglichen. | `true` |
| `allowedCSIDrivers` | Trident verwendet keine kurzlebigen CSI-Inline-Volumes. | Leer |
| `allowedCapabilities` | Für Trident Container ohne Privilegien sind nicht mehr Funktionen erforderlich als für die Standardwerte. Privilegierte Container erhalten alle möglichen Funktionen. | Leer |
| `allowedFlexVolumes` | Trident verwendet keinen "FlexVolume-Treiber", daher sind sie nicht in der Liste der erlaubten Volumes enthalten. | Leer |

| Feld | Beschreibung | Standard |
| --- | --- | --- |
| `allowedHostPaths` | Der Trident-Node-Pod hängt das Root-Dateisystem des Node zusammen, daher bietet es keinen Vorteil, diese Liste zu setzen. | Leer |
| `allowedProcMountTypes` | Trident verwendet keine `ProcMountTypes`. | Leer |
| `allowedUnsafeSysctls` | Trident erfordert keine unsicheren `sysctls`. | Leer |
| `defaultAddCapabilities` | Zu privilegierten Containern müssen keine Funktionen hinzugefügt werden. | Leer |
| `defaultAllowPrivilegeEscalation` | In jedem Trident Pod werden Berechtigungen erteilt. | `false` |
| `forbiddenSysctls` | Nein `sysctls` sind zulässig. | Leer |
| `fsGroup` | Trident Container werden als Root ausgeführt. | `RunAsAny` |
| `hostIPC` | Für das Mounten von NFS-Volumes ist Host-IPC zur Kommunikation mit erforderlich `nfsd` | `true` |
| `hostNetwork` | Iscsiadm erfordert, dass das Hostnetzwerk mit dem iSCSI-Daemon kommunizieren kann. | `true` |
| `hostPID` | Host-PID ist erforderlich, um zu überprüfen, ob `rpc-statd` auf dem Knoten ausgeführt wird. | `true` |
| `hostPorts` | Trident verwendet keine Host Ports. | Leer |
| `privileged` | Trident Node-Pods müssen einen privilegierten Container ausführen, um Volumes mounten zu können. | `true` |
| `readOnlyRootFilesystem` | Trident Node-Pods müssen in das Node-Dateisystem schreiben. | `false` |
| `requiredDropCapabilities` | Trident Node-Pods führen einen privilegierten Container aus und können Funktionen nicht ablegen. | `none` |
| `runAsGroup` | Trident Container werden als Root ausgeführt. | `RunAsAny` |
| `runAsUser` | Trident Container werden als Root ausgeführt. | `runAsAny` |
| `runtimeClass` | Trident verwendet nicht `RuntimeClasses` . | Leer |

| Feld | Beschreibung | Standard |
| --- | --- | --- |
| `seLinux` | Trident ist nicht festgelegt `seLinuxOptions`, da es derzeit Unterschiede gibt, wie Container-Laufzeiten und Kubernetes-Distributionen SELinux handhaben. | Leer |
| `supplementalGroups` | Trident Container werden als Root ausgeführt. | `RunAsAny` |
| `volumes` | Trident Pods erfordern diese Volume-Plug-ins. | `hostPath, projected, emptyDir` |

## Sicherheitskontexteinschränkungen (SCC)

| Etiketten | Beschreibung | Standard |
| --- | --- | --- |
| `allowHostDirVolumePlugin` | Trident-Node-Pods mounten das Root-Dateisystem des Node. | `true` |
| `allowHostIPC` | Für das Mounten von NFS-Volumes muss Host IPC mit kommunizieren `nfsd`. | `true` |
| `allowHostNetwork` | Iscsiadm erfordert, dass das Hostnetzwerk mit dem iSCSI-Daemon kommunizieren kann. | `true` |
| `allowHostPID` | Host-PID ist erforderlich, um zu überprüfen, ob `rpc-statd` auf dem Knoten ausgeführt wird. | `true` |
| `allowHostPorts` | Trident verwendet keine Host Ports. | `false` |
| `allowPrivilegeEscalation` | Privilegierte Container müssen die Eskalation von Berechtigungen ermöglichen. | `true` |
| `allowPrivilegedContainer` | Trident Node-Pods müssen einen privilegierten Container ausführen, um Volumes mounten zu können. | `true` |
| `allowedUnsafeSysctls` | Trident erfordert keine unsicheren `sysctls`. | `none` |
| `allowedCapabilities` | Für Trident Container ohne Privilegien sind nicht mehr Funktionen erforderlich als für die Standardwerte. Privilegierte Container erhalten alle möglichen Funktionen. | Leer |
| `defaultAddCapabilities` | Zu privilegierten Containern müssen keine Funktionen hinzugefügt werden. | Leer |
| `fsGroup` | Trident Container werden als Root ausgeführt. | `RunAsAny` |

| Etiketten | Beschreibung | Standard |
| --- | --- | --- |
| `groups` | Dieses SCC ist speziell für Trident bestimmt und an den Anwender gebunden. | Leer |
| `readOnlyRootFilesystem` | Trident Node-Pods müssen in das Node-Dateisystem schreiben. | `false` |
| `requiredDropCapabilities` | Trident Node-Pods führen einen privilegierten Container aus und können Funktionen nicht ablegen. | `none` |
| `runAsUser` | Trident Container werden als Root ausgeführt. | `RunAsAny` |
| `seLinuxContext` | Trident ist nicht festgelegt `seLinuxOptions`, da es derzeit Unterschiede gibt, wie Container-Laufzeiten und Kubernetes-Distributionen SELinux handhaben. | Leer |
| `seccompProfiles` | Privilegierte Container laufen immer „unbegrenzt“. | Leer |
| `supplementalGroups` | Trident Container werden als Root ausgeführt. | `RunAsAny` |
| `users` | Es ist ein Eintrag verfügbar, um diesen SCC an den Trident-Benutzer im Trident Namespace zu binden. | k. A. |
| `volumes` | Trident Pods erfordern diese Volume-Plug-ins. | `hostPath, downwardAPI, projected, emptyDir` |