Referenz

Astra Trident

NetApp
April 03, 2024

Inhalt

Referenz

Astra Trident-Ports

Erfahren Sie mehr über die Kommunikationsports von Astra Trident.

Astra Trident-Ports

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

Astra Trident REST-API

Während "Tridentctl-Befehle und -Optionen" Die einfachste Möglichkeit, mit der Astra Trident REST-API zu interagieren, können Sie den REST-Endpunkt direkt verwenden, wenn Sie es bevorzugen.

Wann die REST-API verwendet werden soll

REST-API ist nützlich für erweiterte Installationen, in denen Astra Trident als eigenständige Binärdatei in Implementierungen ohne Kubernetes genutzt wird.

Für höhere Sicherheit bietet der Astra Trident an `REST API` Ist standardmäßig auf localhost beschränkt, wenn in einem Pod ausgeführt wird. Um dieses Verhalten zu ändern, müssen Sie Astra Trident's einstellen `-address` Argument in seiner Pod-Konfiguration.

REST-API wird verwendet

Für Beispiele, wie diese APIs aufgerufen werden, geben Sie das Debug (`-d`) Flagge. Weitere Informationen finden Sie unter "Managen Sie Astra 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 Astra 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 Astra Trident mit tridentctl".

Befehlszeilenoptionen

Astra Trident stellt verschiedene Befehlszeilenoptionen für den Trident Orchestrator bereit. 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 die Kubernetes-Unterstützung 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 die Kubernetes-Unterstützung 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 implementiert werden; es unterstützt jedoch nur unsichere Verbindungen zum API-Server. Mit der können Sie Trident sicher in einem Pod implementieren `-k8s_pod` Option.

Docker

`-volume_driver <name>`

Treibername, der bei der Registrierung des Docker-Plug-ins verwendet wird. Standardmäßig auf `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. Nutzung `-address ""` Damit die REST-Schnittstelle über die POD-IP-Adresse zugänglich ist.

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 `PersistentVolumeClaim` Anforderung eines neuen `PersistentVolume` Einer bestimmten Größe von einem Kubernetes aus `StorageClass` Das wurde zuvor vom Administrator konfiguriert.

2. Kubernetes `StorageClass` Identifiziert Trident als seine bereitstellung und enthält Parameter, die Trident zur Bereitstellung eines Volumes für die angeforderte Klasse angeben.
3. Trident sieht seinen eigenen Blick `StorageClass` Mit dem gleichen Namen, der die Übereinstimmung identifiziert `Backends` Und `StoragePools` Die sie für die Bereitstellung von Volumes für die Klasse einsetzen kann.
4. Trident stellt Storage auf einem passenden Back-End bereit und erstellt zwei Objekte: A `PersistentVolume` In Kubernetes informiert Kubernetes über das Finden, Mounten und behandeln des Volumes und ein Volume in Trident, das die Beziehung zwischen den beibehält `PersistentVolume` Und dem tatsächlichen Storage.
5. Kubernetes bindet das `PersistentVolumeClaim` Zum neuen `PersistentVolume`. Pods, die die enthalten `PersistentVolumeClaim` Mounten Sie dieses PersistenzVolume auf jedem Host, auf dem es ausgeführt wird.
6. Ein Benutzer erstellt ein `VolumeSnapshot` Eines vorhandenen PVC unter Verwendung eines `VolumeSnapshotClass` Das verweist auf Trident.
7. Trident identifiziert das dem PVC zugeordnete Volume und erstellt einen Snapshot des Volumes auf dem Back-End. Es erzeugt auch ein `VolumeSnapshotContent` Damit wird Kubernetes angewiesen, den Snapshot zu identifizieren.
8. Ein Benutzer kann ein erstellen `PersistentVolumeClaim` Wird verwendet `VolumeSnapshot` Als Quelle.
9. Trident identifiziert den erforderlichen Snapshot und führt die gleichen Schritte aus, die bei der Erstellung eines erforderlich sind `PersistentVolume` Und A `Volume`.

Für weitere Informationen über Kubernetes-Objekte empfehlen wir Ihnen, die zu lesen "Persistente Volumes" Der Kubernetes-Dokumentation.

Kubernetes `PersistentVolumeClaim` Objekte

Ein Kubernetes `PersistentVolumeClaim` Objekt ist eine Storage-Anfrage von einem Kubernetes Cluster-Benutzer.

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-Files, gcp-cim, 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

Anmerkung	Volume-Option	Unterstützte Treiber
trident.netapp.io/snapshotPolicy	SnapshotPolicy	ontap-nas ontap-nas-Economy, ontap-nas-flexgroup, ontap-san
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 den verfügt `Delete` Rückgewinnungsrichtlinie: Trident löscht sowohl das PV als auch das Backvolume, 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 den verwendet `Retain` Richtlinie: Trident ignoriert es und geht davon aus, dass der Administrator die Datei über Kubernetes und das Backend bereinigt, damit 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. Sie finden sie hier `On-Demand Volume Snapshots` Um zu sehen, wie das funktionieren würde.

Trident enthält außerdem die `cloneFromPVC` Und `splitOnClone` Anmerkungen 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 ein PVC aufgerufen hat `mysql`, Der Benutzer kann ein neues PVC mit dem Namen erstellen `mysqlclone` Durch die Verwendung der Anmerkung, z. B. `trident.netapp.io/cloneFromPVC: mysql`. 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.
- Mit dem `ontap-nas` Und `ontap-san` Treiber, kann es wünschenswert sein, die PVC-Anmerkung zu setzen `trident.netapp.io/splitOnClone` Zusammen mit `trident.netapp.io/cloneFromPVC`. Mit `trident.netapp.io/splitOnClone` Auf einstellen `true`, Trident teilt das geklonte Volume vom übergeordneten Volume auf und sorgt so für eine vollständige Entkopplung des geklonten Volume vom übergeordneten Volume – und zwar auf Kosten des Verlusts von Storage-Effizienz. Keine Einstellung `trident.netapp.io/splitOnClone` Oder auf einstellen `false` Dies senkt den Platzbedarf im Back-

End. Dies verursacht Abhängigkeiten zwischen dem übergeordneten und den Klon-Volumes, sodass das übergeordnete Volume nur gelöscht werden kann, wenn der Klon zuvor gelöscht wird. 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.

Der `sample-input` Das Verzeichnis enthält Beispiele für PVC-Definitionen zur Verwendung mit Trident. Siehe Eine vollständige Beschreibung der Parameter und Einstellungen zu Trident Volumes.

Kubernetes `PersistentVolume` Objekte

Ein Kubernetes `PersistentVolume` Objekt stellt eine Storage-Komponente dar, die dem Kubernetes-Cluster zur Verfügung gestellt wird. Es weist einen Lebenszyklus auf, der unabhängig vom POD ist, der ihn nutzt.

Trident erstellt `PersistentVolume` Objekte werden beim Kubernetes Cluster automatisch auf Basis der Volumes registriert, die bereitgestellt werden. Sie sollten diese nicht selbst verwalten.

Wenn Sie eine PVC erstellen, die sich auf eine Trident-basierte bezieht `StorageClass`, Trident stellt ein neues Volume anhand der entsprechenden Storage-Klasse 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.

Kubernetes `StorageClass` Objekte

Kubernetes `StorageClass` Objekte werden in mit Namen angegeben `PersistentVolumeClaims` So stellen Sie Speicher mit einer Reihe von Eigenschaften bereit. 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 so 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 Backend-Namen zu Listen Storage-Pools in NetApp zu nutzen
Zusätzlich StoragePools	Zuordnen[String]StringList	Nein	Zuordnung der Backend-Namen Listen von Speicherpools in
Unter Ausnahme von StoragePools	Zuordnen[String]StringList	Nein	Zuordnung der Backend-Namen zu Listen der Speicherpools in

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

Attribut	Typ	Werte	Angebot	Anfrage	Unterstützt von
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
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.

Im Idealfall können Sie verwenden `attributes` Um die Eigenschaften des Storage zu modellieren, können Sie die Anforderungen einer bestimmten Klasse erfüllen. Trident erkennt und wählt automatisch Storage Pools aus, die mit *all* der übereinstimmen `attributes` Die Sie angeben.

Wenn Sie feststellen, dass Sie nicht in der Lage sind, zu verwenden `attributes` Um automatisch die richtigen Pools für eine Klasse auszuwählen, können Sie die verwenden `storagePools` Und `additionalStoragePools` Parameter zur weiteren Verfeinerung der Pools oder sogar zur Auswahl einer bestimmten Gruppe von Pools.

Sie können das verwenden `storagePools` Parameter zur weiteren Einschränkung des Pools, die mit den angegebenen übereinstimmen `attributes`. Mit anderen Worten: Trident verwendet die Schnittstelle von Pools, die vom identifiziert werden `attributes` Und `storagePools` Parameter für die Bereitstellung. Sie können entweder allein oder beides zusammen verwenden.

Sie können das verwenden `additionalStoragePools` Parameter zur Erweiterung des Pools, die Trident für die Bereitstellung verwendet, unabhängig von den vom ausgewählten Pools `attributes` Und `storagePools` Parameter.

Sie können das verwenden `excludeStoragePools` Parameter zum Filtern des Pools, den Trident für die Bereitstellung verwendet. Mit diesem Parameter werden alle Pools entfernt, die übereinstimmen.

Im `storagePools` Und `additionalStoragePools` Parameter, jeder Eintrag nimmt das Formular `<backend>:<storagePoolList>`, Wo `<storagePoolList>` Ist eine kommagetrennte Liste von Speicherpools für das angegebene Backend. Beispiel: Ein Wert für `additionalStoragePools` Könnte 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. Verwenden Sie können `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 usw.	Der Dateisystemtyp 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 und höher

VolumeBindingmodus	Zeichenfolge	Sofort, WaitForFirstConsumer	Legen Sie fest, wann Volume Binding und dynamische Bereitstellung stattfindet	Alle	1.19 - 1.26

- Der `fsType` Parameter wird verwendet, um den gewünschten Filesystem-Typ für SAN-LUNs zu steuern. Darüber hinaus verwendet Kubernetes auch Präsenz von `fsType` In einer Speicherklasse, die darauf hinweist, dass ein Dateisystem vorhanden ist. Das Volume-Eigentum kann über den gesteuert werden `fsGroup` Sicherheitskontext eines Pods nur wenn `fsType` Ist festgelegt. Siehe "Kubernetes: Einen Sicherheitskontext für einen Pod oder Container konfigurieren" Für eine Übersicht über die Einstellung des Volume-Besitzes mit dem `fsGroup` Kontext. Kubernetes wendet das an `fsGroup` Wert nur, 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. Zur Verwendung `fsGroup` Die Storage-Klasse muss noch ein angeben `fsType`. Sie können es auf einstellen `nfs` Oder ein nicht-Null-Wert.
- Siehe "Erweitern Sie Volumes" Für weitere Informationen zur Volume-Erweiterung.
- Das Trident Installationspaket bietet verschiedene Beispiele für Storage-Klassen, die mit Trident in verwendet werden können `sample-input/storage-class-*.yaml`. Durch das Löschen einer Kubernetes-Storage-Klasse wird auch die entsprechende Trident-Storage-Klasse gelöscht.

Kubernetes `VolumeSnapshotClass` Objekte

Kubernetes `VolumeSnapshotClass` Objekte sind analog `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.

A `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, dass Volume-Snapshots von anfordert `csi-snapclass` Die Klasse werden von Trident übernommen. Der `deletionPolicy` Gibt die Aktion an, die ausgeführt werden soll, wenn ein Snapshot gelöscht werden muss. Wenn `deletionPolicy` Ist auf festgelegt `Delete`, Die Volume-Snapshot-Objekte sowie der zugrunde liegende Snapshot auf dem Storage-Cluster werden entfernt, wenn ein Snapshot

gelöscht wird. Alternativ können Sie ihn auf einstellen `Retain` Bedeutet das `VolumeSnapshotContent` Und der physische Snapshot wird beibehalten.

Kubernetes `VolumeSnapshot` 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.

Sobald eine Volume Snapshot-Anfrage eingeht, managt Trident automatisch die Erstellung des Snapshots für das Volume auf dem Backend und legt den Snapshot offen, indem er einen eindeutigen erstellt `VolumeSnapshotContent` Objekt: 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.

Kubernetes `VolumeSnapshotContent` Objekte

Ein Kubernetes `VolumeSnapshotContent` Objekt stellt einen Snapshot dar, der von einem bereits bereitgestellten Volume entnommen wurde. Es ist analog zu einem `PersistentVolume` Und bedeutet einen bereitgestellten Snapshot auf dem Storage-Cluster. Ähnlich `PersistentVolumeClaim` Und `PersistentVolume` Objekte, wenn ein Snapshot erstellt wird, das `VolumeSnapshotContent` Objekt verwaltet eine 1:1-Zuordnung zum `VolumeSnapshot` Objekt, das die Snapshot-Erstellung angefordert hatte.

Der `VolumeSnapshotContent` Das Objekt enthält Details, die den Snapshot eindeutig identifizieren, z. B. den `snapshotHandle`. Das `snapshotHandle` Ist eine einzigartige Kombination aus dem Namen des PV und dem Namen des `VolumeSnapshotContent` Objekt:

Wenn eine Snapshot-Anfrage eingeht, erstellt Trident den Snapshot auf dem Back-End. Nach der Erstellung des Snapshots konfiguriert Trident einen `VolumeSnapshotContent` Objekt-Storage erstellt und damit den Snapshot der Kubernetes API zur Verfügung gestellt.

> In der Regel müssen Sie das nicht verwalten `VolumeSnapshotContent` Objekt: Eine Ausnahme ist, wann Sie möchten "Importieren Sie einen Volume-Snapshot" Erstellt außerhalb von Astra Trident.

Kubernetes `CustomResourceDefinition` 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 erhalten diese Ressourcen-Definitionen, 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.

Astra Trident verwendet `CustomResourceDefinition` Objekte zur Wahrung der Identität von Trident

Objekten, wie Trident Back-Ends, Trident Storage-Klassen und Trident Volumes. 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. Wenn ein Backend mit erstellt wird, ist das ein einfaches Beispiel `tridentctl`, Eine entsprechende `tridentbackends` Das CRD-Objekt wird 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.
- Bei der Deinstallation von Trident mit dem `tridentctl uninstall` Befehl, Trident Pods werden gelöscht, die erstellten CRDs werden jedoch nicht bereinigt. Siehe "Deinstallieren Sie Trident" Um zu erfahren, wie Trident vollständig entfernt und von Grund auf neu konfiguriert werden kann

Astra Trident `StorageClass` Objekte

Trident erstellt passende Storage-Klassen für Kubernetes `StorageClass` Objekte, die angeben `csi.trident.netapp.io` In ihrem Feld für die bereitstellung. Der Name der Storage-Klasse stimmt mit der der von Kubernetes überein `StorageClass` Objekt, das es repräsentiert.

Mit Kubernetes werden diese Objekte automatisch bei einem Kubernetes erstellt `StorageClass` Und Trident ist für die bereitstellung registriert.

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 werden sie jedoch bei der Registrierung von neuem Kubernetes erstellt `StorageClass` Objekte:

Back-End-Objekte für Astra Trident

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 Kubernetes `StorageClass` Objekt:

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

Astra Trident `StoragePool` Objekte

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.

Astra Trident `Volume` Objekte

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

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

Trident generiert `internalName` Beim Erstellen des Volumes. Dies besteht aus zwei Schritten. Zuerst wird das Speicherpräfix (entweder der Standard) voreingestellt `trident` Oder das Präfix in der Backend-Konfiguration) zum Volume-Namen, was zu einem Namen des Formulars führt `<prefix>-<volume-name>`. Anschließend wird der Name desinfiziert und die im Backend nicht zulässigen Zeichen ersetzt. Bei ONTAP Back-Ends werden Bindestriche mit Unterstriche ersetzt (d. h., der interne Name wird aus `<prefix>_<volume-name>`). Bei Element-Back-Ends werden Unterstriche durch Bindestriche ersetzt.

Sie können Volume-Konfigurationen verwenden, um Volumes direkt über DIE REST-API bereitzustellen. In Kubernetes-Implementierungen gehen die meisten Benutzer jedoch davon aus, den Standard Kubernetes zu verwenden `PersistentVolumeClaim` Methode. Trident erstellt dieses Volume-Objekt automatisch im Rahmen der Bereitstellung
Prozess.

Astra Trident `Snapshot` Objekte

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

Beide `Snapshot` Objekt enthält die unten 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

Attribut	Typ	Erforderlich	Beschreibung
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.

Wenn ein Kubernetes `VolumeSnapshot` Objektanforderung wird erstellt. Trident erstellt ein Snapshot-Objekt auf dem zugrunde gelegten Storage-System. Der `internalName` Dieses Snapshot-Objekt wird durch Kombination des Präfixes generiert `snapshot-` Mit dem `UID` Des `VolumeSnapshot` Objekt (z. B. `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` Und `volumeInternalName` Werden ausgefüllt, indem die Details des Backing abgerufen werden
Datenmenge:

Astra Trident `ResourceQuota` Objekt

Das Trident-Eintreten verbraucht einen `system-node-critical` Priority Class – die in Kubernetes verfügbare Class mit höchster Priorität, damit Astra Trident Volumes beim ordnungsgemäßen Shutdown von Nodes identifizieren und bereinigen kann und Trident Demonset-Pods zulassen kann, dass Workloads mit niedriger Priorität in Clustern mit hohen Ressourcenbelastungen vorbeugen.

Astra Trident setzt hierfür ein `ResourceQuota` Möchten Sie sicherstellen, dass eine „System-Node-kritische“ Prioritätsklasse auf dem Trident-Demonset erfüllt ist. Vor der Implementierung und der Erstellung von Dämonen sucht Astra Trident die `ResourceQuota` Objekt und, falls nicht erkannt, wendet es an.

Wenn Sie mehr Kontrolle über das standardmäßige Ressourcenkontingent und die Prioritätsklasse benötigen, können Sie ein generieren `custom.yaml` Oder konfigurieren Sie die `ResourceQuota` Objekt mit Helm-Diagramm.

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 Ressourcenkontingenten finden Sie unter "Kubernetes: Ressourcenkontingente".

Bereinigung `ResourceQuota` Wenn die Installation fehlschlägt

In seltenen Fällen, in denen die Installation nach dem fehlschlägt `ResourceQuota` Das Objekt wird erstellt, versuchen Sie es zuerst "Deinstallation" Und installieren Sie dann neu.

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

Entfernen `ResourceQuota`

Wenn Sie die eigene Ressourcenzuweisung steuern möchten, können Sie den Astra Trident entfernen `ResourceQuota` Objekt mit dem Befehl:

```
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. Zur Bereitstellung dieser Anpassung ermöglicht Astra Trident während der Installation bestimmte Berechtigungen. In den folgenden Abschnitten werden die Berechtigungen von Astra Trident erläutert.

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	Für den iSCSI-Daemon erforderlich. `iscsiadm` Managt iSCSI-Mounts und verwendet Host-Netzwerke für die Kommunikation mit dem iSCSI-Daemon.
Host-IPC	NFS nutzt Prozesskommunikation (IPC) mit dem NFSD.
Host-PID	Muss gestartet werden `rpc-statd` Für NFS. Astra Trident fragt die Host-Prozesse ab, um festzustellen, ob `rpc-statd` Wird vor dem Mounten von NFS-Volumes ausgeführt.

Berechtigung	Beschreibung
Sorgen	Der `SYS_ADMIN` Diese Funktion wird als Teil der Standardfunktionen für privilegierte Container bereitgestellt. Docker legt beispielsweise die folgenden Funktionen für privilegierte Container fest: `CapPrm: 0000003fffffffff` `CapEff: 0000003fffffffff`
Abt	Seccomp-Profil ist in privilegierten Containern immer „unbegrenzt“; daher kann es in Astra Trident nicht aktiviert werden.
SELinux	Auf OpenShift werden privilegierte Container im betrieben `spc_t` („Super Privileged Container“)-Domain, und unprivilegierte Container werden im ausgeführt `container_t` Domäne. Ein `containerd`, Mit `container-selinux` Installiert, alle Container werden im ausgeführt `spc_t` Domain, die SELinux effektiv deaktiviert. Aus diesem Grund wird Astra Trident nicht hinzugefügt `seLinuxOptions` Zusammen mit Containern.
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-…“. Wenn dies geschieht, prüfen Sie, ob die Namespace-Bezeichnung für verwendet wird `privileged` Wurde geändert. Falls ja, installieren Sie Trident neu.

Pod-Sicherheitsrichtlinien (PSP)

Feld	Beschreibung	Standard
`allowPrivilegeEscalation`	Privilegierte Container müssen die Eskalation von Berechtigungen ermöglichen.	`true`

Feld	Beschreibung	Standard
`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 kein a "FlexVolume-Treiber", Sie sind daher nicht in die Liste der zulässigen Volumen.	Leer
`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 Unsicherheit `sysctls`.	Leer
`defaultAddCapabilities`	Zu privilegierten Containern müssen keine Funktionen hinzugefügt werden.	Leer
`defaultAllowPrivilegeEscalation`	In jedem Trident Pod werden Berechtigungen erteilt.	`false`
`forbiddenSysctls`	Nein `sysctls` Zulässig.	Leer
`fsGroup`	Trident Container werden als Root ausgeführt.	`RunAsAny`
`hostIPC`	Das Mounten von NFS-Volumes erfordert die Kommunikation zwischen dem Host IPC und dem `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` Wird auf dem Node ausgeführt.	`true`
`hostPorts`	Trident verwendet keine Host Ports.	Leer
`privileged`	Trident Node-Pods müssen einen privilegierten Container ausführen, um Volumes mounten zu können.	`true`

Feld	Beschreibung	Standard
`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 wird nicht verwendet `RuntimeClasses`.	Leer
`seLinux`	Trident ist nicht eingerichtet `seLinuxOptions` Weil es derzeit Unterschiede hinsichtlich der Handhabung von Container-Laufzeiten und Kubernetes-Distributionen für SELinux gibt.	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`	Das Mounten von NFS-Volumes erfordert die Kommunikation zwischen dem Host IPC und dem `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` Wird auf dem Node ausgeführt.	`true`
`allowHostPorts`	Trident verwendet keine Host Ports.	`false`
`allowPrivilegeEscalation`	Privilegierte Container müssen die Eskalation von Berechtigungen ermöglichen.	`true`

Etiketten	Beschreibung	Standard
`allowPrivilegedContainer`	Trident Node-Pods müssen einen privilegierten Container ausführen, um Volumes mounten zu können.	`true`
`allowedUnsafeSysctls`	Trident erfordert keine Unsicherheit `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`
`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 eingerichtet `seLinuxOptions` Weil es derzeit Unterschiede hinsichtlich der Handhabung von Container-Laufzeiten und Kubernetes-Distributionen für SELinux gibt.	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`