Referenz

Trident

NetApp
January 15, 2026

Inhalt

Referenz

Trident Häfen

Erfahren Sie mehr über die Anschlüsse, die Trident zur Kommunikation nutzt.

Trident Häfen

Trident verwendet die folgenden Ports für die Kommunikation innerhalb von Kubernetes:

Hafen	Zweck
8443	Backchannel HTTPS
8001	Prometheus-Metriken-Endpunkt
8000	Trident REST-Server
17546	Liveness-/Readiness-Probe-Port, der von Trident -DaemonSet-Pods verwendet wird

Der Anschluss für die Lebendigkeits-/Bereitschaftsprüfung kann während der Installation mithilfe des `--probe-port` Flagge. Es ist wichtig sicherzustellen, dass dieser Port nicht von einem anderen Prozess auf den Worker-Knoten verwendet wird.

Trident REST-API

Während"tridentctl-Befehle und Optionen" Dies ist die einfachste Möglichkeit, mit der Trident REST API zu interagieren; Sie können aber auch direkt den REST-Endpunkt verwenden, wenn Sie dies bevorzugen.

Wann sollte die REST-API verwendet werden?

Die REST-API ist nützlich für fortgeschrittene Installationen, die Trident als eigenständige Binärdatei in Nicht-Kubernetes-Bereitstellungen verwenden.

Für mehr Sicherheit, das Trident `REST API` Beim Ausführen innerhalb eines Pods ist die Ausführung standardmäßig auf localhost beschränkt. Um dieses Verhalten zu ändern, müssen Sie die Einstellungen von Trident anpassen. `-address` Argument in seiner Pod-Konfiguration.

Verwendung der REST-API

Beispiele für den Aufruf dieser APIs erhalten Sie durch Übergeben des Debug-Logs.(`-d`) Flagge. Weitere Informationen finden Sie unter"Trident mit tridentctl verwalten" .

Die API funktioniert wie folgt:

ERHALTEN

`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 angegebenen Objekts ab.

POST

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

Erzeugt ein Objekt des angegebenen Typs.

- Für die Erstellung des Objekts ist eine JSON-Konfiguration erforderlich. Die Spezifikation der einzelnen Objekttypen finden Sie unter"Trident mit tridentctl verwalten" .
- Existiert das Objekt bereits, verhält es sich unterschiedlich: Die Backends aktualisieren das bestehende Objekt, während bei allen anderen Objekttypen der Vorgang fehlschlägt.

LÖSCHEN

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

Löscht die angegebene Ressource.

> Mit Backends oder Speicherklassen verknüpfte Volumes bleiben bestehen; diese müssen separat gelöscht werden. Weitere Informationen finden Sie unter"Trident mit tridentctl verwalten" .

Befehlszeilenoptionen

Trident stellt mehrere Befehlszeilenoptionen für den Trident -Orchestrator bereit. Sie können diese Optionen nutzen, um Ihre Bereitstellung anzupassen.

Protokollierung

`-debug`

Aktiviert die Debug-Ausgabe.

`-loglevel <level>`

Legt den Protokollierungsgrad fest (debug, info, warn, error, fatal). Standardmäßig wird die Info-Seite angezeigt.

Kubernetes

`-k8s_pod`

Nutzen Sie diese Option oder `-k8s_api_server` um die Kubernetes-Unterstützung zu aktivieren. Durch diese Einstellung verwendet Trident die Anmeldeinformationen des Kubernetes-Dienstkontos des übergeordneten Pods, um den API-Server zu kontaktieren. Dies funktioniert nur, wenn Trident als Pod in einem Kubernetes-Cluster mit aktivierten Service Accounts ausgeführt wird.

-k8s_api_server <insecure-address:insecure-port>

Nutzen Sie diese Option oder `-k8s_pod` um die Kubernetes-Unterstützung zu aktivieren. Wenn angegeben, stellt Trident über die bereitgestellte unsichere Adresse und den Port eine Verbindung zum Kubernetes-API-Server her. Dies ermöglicht den Einsatz von Trident außerhalb eines Pods; allerdings werden dabei nur unsichere Verbindungen zum API-Server unterstützt. Um eine sichere Verbindung herzustellen, stellen Sie Trident in einem Pod mit dem bereit. `-k8s_pod` Option.

Docker

-volume_driver <name>

Name des Treibers, der bei der Registrierung des Docker-Plugins verwendet wird. Standardmäßig `netapp` .

-driver_port <port-number>

Lauschen Sie an diesem Port anstatt an einem UNIX-Domain-Socket.

-config <file>

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

AUSRUHEN

-address <ip-or-host>

Gibt die Adresse an, an der der REST-Server von Trident lauschen soll. Standardmäßig wird localhost verwendet. Beim Lauschen auf localhost und Ausführen innerhalb eines Kubernetes-Pods ist die REST-Schnittstelle von außerhalb des Pods nicht direkt zugänglich. Verwenden `-address ""` um die REST-Schnittstelle von der Pod-IP-Adresse aus zugänglich zu machen.

> Die Trident REST-Schnittstelle kann so konfiguriert werden, dass sie nur unter 127.0.0.1 (für IPv4) oder [::1] (für IPv6) lauscht und Anfragen beantwortet.

-port <port-number>

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

-rest

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

Kubernetes- und Trident Objekte

Sie können mit Kubernetes und Trident über REST-APIs interagieren, indem Sie Ressourcenobjekte lesen und schreiben. Es gibt mehrere Ressourcenobjekte, die die Beziehung zwischen Kubernetes und Trident, Trident und Speicher sowie Kubernetes und Speicher festlegen. Einige dieser Objekte werden über Kubernetes verwaltet, die anderen über Trident.

Wie interagieren die Objekte miteinander?

Am einfachsten lassen sich die Objekte, ihr Zweck und ihre Interaktion verstehen, indem man eine einzelne Speicheranfrage eines Kubernetes-Benutzers verfolgt:

1. Ein Benutzer erstellt einen `PersistentVolumeClaim` eine neue anfordern `PersistentVolume` einer bestimmten Größe aus einem Kubernetes `StorageClass` das zuvor vom Administrator konfiguriert wurde.
2. Kubernetes `StorageClass` identifiziert Trident als seinen Provisioner und enthält Parameter, die Trident mitteilen, wie ein Volume für die angeforderte Klasse bereitgestellt werden soll.
3. Trident betrachtet sich selbst `StorageClass` mit demselben Namen, der die Übereinstimmung identifiziert `Backends` Und `StoragePools` dass es zur Bereitstellung von Datenträgern für die Klasse verwendet werden kann.
4. Trident stellt Speicherplatz auf einem passenden Backend bereit und erstellt zwei Objekte: ein `PersistentVolume` in Kubernetes, das Kubernetes mitteilt, wie das Volume gefunden, eingebunden und behandelt wird, und ein Volume in Trident , das die Beziehung zwischen dem Volume beibehält. `PersistentVolume` und der eigentliche Speicher.
5. Kubernetes bindet die `PersistentVolumeClaim` zum Neuen `PersistentVolume` . Kapseln, die Folgendes beinhalten `PersistentVolumeClaim` Mounten Sie dieses PersistentVolume auf jedem Host, auf dem es ausgeführt wird.
6. Ein Benutzer erstellt einen `VolumeSnapshot` eines vorhandenen PVC, unter Verwendung eines `VolumeSnapshotClass` Das deutet auf Trident hin.
7. Trident identifiziert das mit dem PVC verknüpfte Volume und erstellt im Backend einen Snapshot des Volumes. Es erzeugt auch ein `VolumeSnapshotContent` Das weist Kubernetes an, wie der Snapshot identifiziert werden kann.
8. Ein Benutzer kann einen erstellen `PersistentVolumeClaim` mit `VolumeSnapshot` als Quelle.
9. Trident identifiziert den erforderlichen Snapshot und führt dieselben Schritte aus, die auch beim Erstellen eines Snapshots erforderlich sind. `PersistentVolume` und ein `Volume` .

Für weiterführende Informationen zu Kubernetes-Objekten empfehlen wir Ihnen dringend, Folgendes zu lesen: "Persistente Datenträger" Abschnitt der Kubernetes-Dokumentation.

Kubernetes `PersistentVolumeClaim` Objekte

Ein Kubernetes `PersistentVolumeClaim` Ein Objekt ist eine Speicheranforderung, die von einem Kubernetes-Cluster-Benutzer gestellt wird.

Zusätzlich zur Standardspezifikation ermöglicht Trident den Benutzern, die folgenden volumespezifischen Annotationen anzugeben, wenn sie die in der Backend-Konfiguration festgelegten Standardeinstellungen überschreiben möchten:

Anmerkung	Lautstärkeoption	Unterstützte Treiber
trident.netapp.io/fileSystem	Dateisystem	ontap-san, solidfire-san,ontap-san-economy
trident.netapp.io/cloneFromPVC	cloneSourceVolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, gcp-cvs, ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ontap-nas, ontap-san
trident.netapp.io/protocol	Protokoll	beliebig

Anmerkung	Lautstärkeoption	Unterstützte Treiber
trident.netapp.io/exportPolicy	Exportrichtlinie	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	Snapshot-Richtlinie	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotReserve	Snapshot-Reserve	ontap-nas, ontap-nas-flexgroup, ontap-san, gcp-cvs
trident.netapp.io/snapshotDirectory	Snapshot-Verzeichnis	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/blockSize	Blockgröße	solidfire-san

Wenn die erstellte PV die `Delete` Gemäß der Rückforderungsrichtlinie löscht Trident sowohl das PV als auch das zugehörige Datenvolumen, wenn das PV freigegeben wird (d. h. wenn der Benutzer das PVC löscht). Sollte der Löschvorgang fehlschlagen, kennzeichnet Trident den PV entsprechend und wiederholt den Vorgang regelmäßig, bis er erfolgreich ist oder der PV manuell gelöscht wird. Wenn die PV die `Retain` Trident ignoriert diese Richtlinie und geht davon aus, dass der Administrator das Volume aus Kubernetes und dem Backend entfernt, sodass es vor seiner Löschung gesichert oder überprüft werden kann. Beachten Sie, dass das Löschen des PV nicht dazu führt, dass Trident das zugehörige Datenträgervolumen löscht. Sie sollten es mithilfe der REST-API entfernen.(`tridentctl`).

Trident unterstützt die Erstellung von Volume Snapshots mithilfe der CSI-Spezifikation: Sie können einen Volume Snapshot erstellen und ihn als Datenquelle verwenden, um vorhandene PVCs zu klonen. Auf diese Weise können zeitpunktbezogene Kopien von PVs in Form von Snapshots für Kubernetes bereitgestellt werden. Die Snapshots können dann verwendet werden, um neue PVs zu erstellen. Schau dir das mal an `On-Demand Volume Snapshots` um zu sehen, wie das funktionieren würde.

Trident bietet außerdem die `cloneFromPVC` Und `splitOnClone` Anmerkungen zum Erstellen von Klonen. Mithilfe dieser Annotationen können Sie ein PVC klonen, ohne die CSI-Implementierung verwenden zu müssen.

Hier ein Beispiel: Wenn ein Benutzer bereits eine PVC namens besitzt `mysql` Der Benutzer kann eine neue PVC erstellen, die `mysqlclone` durch Verwendung der Annotation, wie zum Beispiel `trident.netapp.io/cloneFromPVC: mysql` . Mit dieser Annotation klont Trident das Volume, das dem MySQL-PVC entspricht, anstatt ein Volume von Grund auf neu zu erstellen.

Beachten Sie folgende Punkte:

- NetApp empfiehlt das Klonen eines ungenutzten Volumes.
- Ein PVC und sein Klon sollten sich im selben Kubernetes-Namespace befinden und dieselbe Storage-Klasse haben.
- Mit der `ontap-nas` Und `ontap-san` Für Fahrer könnte es wünschenswert sein, die PVC-Anmerkung festzulegen. `trident.netapp.io/splitOnClone` in Verbindung mit `trident.netapp.io/cloneFromPVC` . Mit `trident.netapp.io/splitOnClone` eingestellt auf `true` Trident trennt das geklonte Volume vom übergeordneten Volume und entkoppelt so den Lebenszyklus des geklonten Volumes vollständig von seinem übergeordneten Volume, allerdings auf Kosten einer gewissen Speichereffizienz. Keine Einstellung `trident.netapp.io/splitOnClone` oder es auf `false` Dies führt zu einem geringeren Speicherplatzverbrauch im Backend, allerdings auf Kosten

der Schaffung von Abhängigkeiten zwischen dem übergeordneten Volume und dem Klon-Volume, sodass das übergeordnete Volume erst gelöscht werden kann, wenn das Klon-Volume zuvor gelöscht wurde. Ein Szenario, in dem das Aufteilen des Klons sinnvoll ist, ist das Klonen eines leeren Datenbank-Volumes, bei dem zu erwarten ist, dass sich das Volume und sein Klon stark unterscheiden und nicht von den Speichereffizienzen von ONTAP profitieren.

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

Kubernetes `PersistentVolume` Objekte

Ein Kubernetes `PersistentVolume` Das Objekt repräsentiert einen Speicherbereich, der dem Kubernetes-Cluster zur Verfügung gestellt wird. Es hat einen Lebenszyklus, der unabhängig von der Kapsel ist, die es verwendet.

Trident erschafft `PersistentVolume` Objekte werden automatisch im Kubernetes-Cluster registriert, basierend auf den von ihm bereitgestellten Volumes. Es wird nicht von Ihnen erwartet, dass Sie diese selbst verwalten.

Wenn Sie ein PVC erstellen, das sich auf ein Trident-basiertes System bezieht `StorageClass` Trident provisioniert ein neues Volume unter Verwendung der entsprechenden Speicherklasse und registriert ein neues PV für dieses Volume. Bei der Konfiguration des bereitgestellten Volumes und des zugehörigen PV befolgt Trident die folgenden Regeln:

- Trident generiert einen PV-Namen für Kubernetes und einen internen Namen, der zur Bereitstellung des Speichers verwendet wird. In beiden Fällen wird dadurch sichergestellt, dass die Namen in ihrem Geltungsbereich einzigartig sind.
- Das Volumen entspricht so genau wie möglich der gewünschten Größe im PVC, wird aber je nach Plattform gegebenenfalls auf die nächstliegende zuordenbare Menge aufgerundet.

Kubernetes `StorageClass` Objekte

Kubernetes `StorageClass` Objekte werden anhand ihres Namens angegeben in `PersistentVolumeClaims` Speicherplatz mit einer Reihe von Eigenschaften bereitstellen. Die Speicherklasse selbst identifiziert den zu verwendenden Provisionierer und definiert diesen Satz von Eigenschaften in einer für den Provisionierer verständlichen Weise.

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

Ein Kubernetes `StorageClass` Ein 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 geben Trident an, wie Volumes für die Klasse bereitgestellt werden sollen.

Die Parameter der Speicherklasse sind:

Attribut	Typ	Erforderlich	Beschreibung
Attribute	map[string]string	NEIN	Siehe den Abschnitt „Attribute“ unten.
Speicherbecken	map[string]StringList	NEIN	Zuordnung von Backend-Namen zu Listen von Speicherpools innerhalb
zusätzliche Speicherpools	map[string]StringList	NEIN	Zuordnung von Backend-Namen zu Listen von Speicherpools innerhalb
Speicherpools ausschließen	map[string]StringList	NEIN	Zuordnung von Backend-Namen zu Listen von Speicherpools innerhalb

Speicherattribute und ihre möglichen Werte lassen sich in Speicherpool-Auswahlattribute und Kubernetes-Attribute unterteilen.

Auswahlattribute für Speicherpools

Diese Parameter legen fest, welche von Trident verwalteten Speicherpools zur Bereitstellung von Volumes eines bestimmten Typs verwendet werden sollen.

Attribut	Typ	Werte	Angebot	Anfrage	Unterstützt von
media[1]	Schnur	HDD, Hybrid, SSD	Der Pool enthält Medien dieses Typs; hybrid bedeutet beides	Medientyp angegeben	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
Bereitstellungstyp	Schnur	dünn, dick	Pool unterstützt diese Bereitstellungsmethode	Bereitstellungsmethode angegeben	dick: alles vom Fass; dünn: alles vom Fass & Solidfire-San

Attribut	Typ	Werte	Angebot	Anfrage	Unterstützt von
Backend-Typ	Schnur	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, gcp-cvs, azure-netapp-files, ontap-san-economy	Pool gehört zu dieser Art von Backend.	Backend spezifiziert	Alle Fahrer
Momentaufnahmen	bool	wahr, falsch	Pool unterstützt Volumes mit Snapshots	Volume mit aktivierten Snapshots	ontap-nas, ontap-san, solidfire-san, gcp-cvs
Klone	bool	wahr, falsch	Pool unterstützt das Klonen von Volumes	Volume mit aktivierten Klonen	ontap-nas, ontap-san, solidfire-san, gcp-cvs
Verschlüsselung	bool	wahr, falsch	Pool unterstützt verschlüsselte Volumes	Volume mit aktivierter Verschlüsselung	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	int	positive ganze Zahl	Pool ist in der Lage, IOPS in diesem Bereich zu garantieren.	Volumen garantiert diese IOPS	solidfire-san

[1]: Wird von ONTAP Select -Systemen nicht unterstützt.

In den meisten Fällen haben die angeforderten Werte direkten Einfluss auf die Bereitstellung; beispielsweise führt die Anforderung von Thick Provisioning zu einem Thick-Provisioning-Volume. Allerdings verwendet ein Element-Speicherpool seine angebotenen minimalen und maximalen IOPS-Werte zur Festlegung der QoS-Werte anstelle der angeforderten Werte. In diesem Fall wird der angeforderte Wert nur zur Auswahl des Speicherpools verwendet.

Im Idealfall können Sie verwenden `attributes` allein die Eigenschaften des Speichers zu modellieren, die Sie benötigen, um die Bedürfnisse einer bestimmten Klasse zu befriedigen. Trident erkennt und wählt automatisch Speicherpools aus, die *allen* der Anforderung entsprechen. `attributes` die Sie angeben.

Sollten Sie feststellen, dass Sie nicht in der Lage sind, zu nutzen `attributes` Um die richtigen Pools für eine Klasse automatisch auszuwählen, können Sie Folgendes verwenden: `storagePools` Und `additionalStoragePools` Parameter, um die Pools weiter zu verfeinern oder sogar eine bestimmte Gruppe von Pools auszuwählen.

Sie können die `storagePools` Parameter zur weiteren Einschränkung der Menge der Pools, die mit einem bestimmten Parameter übereinstimmen. `attributes` . Mit anderen Worten, Trident nutzt die Schnittmenge der durch die `attributes` Und `storagePools` Parameter für die Bereitstellung. Sie können entweder den einen Parameter einzeln oder beide zusammen verwenden.

Sie können die `additionalStoragePools` Parameter zur Erweiterung der von Trident für die Bereitstellung verwendeten Pools, unabhängig von den vom `attributes` Und `storagePools` Parameter.

Sie können die `excludeStoragePools` Parameter zum Filtern der Pools, die Trident für die Bereitstellung verwendet. Durch die Verwendung dieses Parameters werden alle passenden Pools entfernt.

Im `storagePools` Und `additionalStoragePools` Parameter, jeder Eintrag hat die Form `<backend>:<storagePoolList>` , Wo `<storagePoolList>` ist eine durch Kommas getrennte Liste von Speicherpools für das angegebene Backend. Zum Beispiel ein Wert für `additionalStoragePools` könnte aussehen `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze` . Diese Listen akzeptieren reguläre Ausdrücke sowohl für die Backend- als auch für die Listenwerte. Sie können verwenden `tridentctl get backend` um die Liste der Backends und ihrer Pools zu erhalten.

Kubernetes-Attribute

Diese Attribute haben keinen Einfluss auf die Auswahl von Speicherpools/Backends durch Trident während der dynamischen Bereitstellung. Stattdessen liefern diese Attribute lediglich Parameter, die von Kubernetes Persistent Volumes unterstützt werden. Die Worker-Knoten sind für Dateisystem-Erstellungsoperationen zuständig und benötigen möglicherweise Dateisystem-Dienstprogramme wie xfsprogs.

Attribut	Typ	Werte	Beschreibung	Relevante Treiber	Kubernetes-Version
fsType	Schnur	ext4, ext3, xfs	Der Dateisystemtyp für Blockvolumes	solidfire-san, ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy	Alle
Volumenerweiterung zulassen	boolescher Wert	wahr, falsch	Unterstützung für die Vergrößerung der PVC-Größe aktivieren oder deaktivieren	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, gcp-cvs, azure-netapp-files	1.11+
volumeBindingMode	Schnur	Sofort, Warten auf den ersten Kunden	Wählen Sie den Zeitpunkt für die Volumenbindung und die dynamische Bereitstellung aus.	Alle	1,19 - 1,26

- Der `fsType` Der Parameter dient zur Steuerung des gewünschten Dateisystemtyps für SAN-LUNs. Darüber hinaus nutzt Kubernetes auch die Anwesenheit von `fsType` in einer Speicherklasse, um anzuzeigen, dass ein Dateisystem existiert. Die Volumenbesitzverhältnisse können über die `fsGroup` Sicherheitskontext eines Pods nur wenn `fsType` ist festgelegt. Siehe"Kubernetes: Konfigurieren eines Sicherheitskontexts für einen Pod oder Container" Für eine Übersicht über die Festlegung der Volumenzugehörigkeit mithilfe des `fsGroup` Kontext. Kubernetes wird die `fsGroup` Wert nur dann, wenn:

  - `fsType`wird in der Speicherklasse festgelegt.
  - Der PVC-Zugriffsmodus ist RWO.

  Bei NFS-Speichertreibern ist ein Dateisystem bereits als Teil des NFS-Exports vorhanden. Um zu verwenden `fsGroup` Die Speicherklasse muss noch eine angeben `fsType` Sie können es so einstellen: `nfs` oder ein beliebiger Wert ungleich null.

- Siehe"Volumen erweitern" Weitere Einzelheiten zur Volumenerweiterung finden Sie hier.
- Das Trident -Installationspaket enthält mehrere Beispieldefinitionen für Speicherklassen zur Verwendung mit Trident `.sample-input/storage-class-*.yaml` . Das Löschen einer Kubernetes-Speicherklasse führt dazu, dass auch die entsprechende Trident -Speicherklasse gelöscht wird.

Kubernetes `VolumeSnapshotClass` Objekte

Kubernetes `VolumeSnapshotClass` Objekte sind analog zu `StorageClasses` . Sie helfen dabei, 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` Um Snapshots zu erstellen, muss ein Administrator dies definieren. 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` legt fest, dass Kubernetes Anfragen für Volume-Snapshots des `csi-snapclass` Die Klassen werden von Trident verwaltet. Der `deletionPolicy` legt fest, welche Aktion ausgeführt werden soll, wenn ein Snapshot gelöscht werden muss. Wann `deletionPolicy` ist eingestellt auf `Delete` Beim Löschen eines Snapshots werden sowohl die Volume-Snapshot-Objekte als auch der zugrunde liegende Snapshot auf dem Speichercluster entfernt. Alternativ können Sie es auch so einstellen: `Retain` bedeutet, dass `VolumeSnapshotContent` und die physische Momentaufnahme wird beibehalten.

Kubernetes `VolumeSnapshot` Objekte

Ein Kubernetes `VolumeSnapshot` Bei „object“ handelt es sich um eine Anfrage zur Erstellung eines

Snapshots eines Volumes. So wie ein PVC eine Anfrage eines Benutzers nach einem Volume darstellt, ist ein Volume-Snapshot eine Anfrage eines Benutzers zur Erstellung eines Snapshots eines bestehenden PVC.

Wenn eine Anfrage für einen Volume-Snapshot eingeht, verwaltet Trident automatisch die Erstellung des Snapshots für das Volume im Backend und stellt den Snapshot durch die Erstellung einer eindeutigen ID bereit.
`VolumeSnapshotContent` Objekt. Sie können Snapshots aus bestehenden PVCs erstellen und diese Snapshots als Datenquelle beim Erstellen neuer PVCs verwenden.

> Der Lebenszyklus eines VolumeSnapshots ist unabhängig vom Quell-PVC: Ein Snapshot bleibt auch dann erhalten, wenn der Quell-PVC gelöscht wird. Beim Löschen eines PVCs mit zugehörigen Snapshots markiert Trident das zugehörige Datenträgervolume im Status **Wird gelöscht**, entfernt es aber nicht vollständig. Das Volume wird entfernt, sobald alle zugehörigen Snapshots gelöscht sind.

Kubernetes `VolumeSnapshotContent` Objekte

Ein Kubernetes `VolumeSnapshotContent` Das Objekt stellt eine Momentaufnahme eines bereits bereitgestellten Volumes dar. Es ist analog zu einem `PersistentVolume` und kennzeichnet einen bereitgestellten Snapshot auf dem Speichercluster. Ähnlich `PersistentVolumeClaim` Und `PersistentVolume` Objekte, wenn ein Snapshot erstellt wird, `VolumeSnapshotContent` Das Objekt verwaltet eine Eins-zu-Eins-Zuordnung zum `VolumeSnapshot` Objekt, das die Erstellung des Snapshots angefordert hatte.

Der `VolumeSnapshotContent` Das Objekt enthält Details, die den Snapshot eindeutig identifizieren, wie zum Beispiel die `snapshotHandle` . Das `snapshotHandle` ist eine einzigartige Kombination aus dem Namen der PV und dem Namen der `VolumeSnapshotContent` Objekt.

Wenn eine Snapshot-Anfrage eingeht, erstellt Trident den Snapshot im Backend. Nachdem der Snapshot erstellt wurde, konfiguriert Trident einen `VolumeSnapshotContent` Das Objekt wird erstellt und somit der Kubernetes-API zugänglich gemacht.

> Normalerweise müssen Sie die `VolumeSnapshotContent` Objekt. Eine Ausnahme hiervon besteht, wenn Sie möchten"einen Volume-Snapshot importieren" außerhalb von Trident erstellt.

Kubernetes `VolumeGroupSnapshotClass` Objekte

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

A `VolumeGroupSnapshotClass` Um eine Gruppe von Snapshots zu erstellen, muss dies von einem Administrator definiert werden. Eine Snapshot-Klasse für eine Volumengruppe wird mit folgender Definition erstellt:

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

Der `driver` legt fest, dass Kubernetes Anfragen für Volume-Gruppen-Snapshots der `csi-group-snap-class` Die Klassen werden von Trident verwaltet. Der `deletionPolicy` legt die Aktion fest, die ausgeführt werden soll, wenn ein Gruppen-Snapshot gelöscht werden muss. Wann `deletionPolicy` ist eingestellt auf `Delete` Beim Löschen eines Snapshots werden sowohl die Volume-Group-Snapshot-Objekte als auch der zugrunde liegende Snapshot auf dem Speichercluster entfernt. Alternativ können Sie es auch so einstellen: `Retain` bedeutet, dass `VolumeGroupSnapshotContent` und die physische Momentaufnahme wird beibehalten.

Kubernetes `VolumeGroupSnapshot` Objekte

Ein Kubernetes `VolumeGroupSnapshot` Das Objekt ist eine Anfrage zur Erstellung eines Snapshots mehrerer Volumes. So wie eine PVC eine Anfrage eines Benutzers nach einem Volume darstellt, ist ein Volume-Gruppen-Snapshot eine Anfrage eines Benutzers zur Erstellung eines Snapshots einer bestehenden PVC.

Wenn eine Anfrage für einen Volume-Gruppen-Snapshot eingeht, verwaltet Trident automatisch die Erstellung des Gruppen-Snapshots für die Volumes im Backend und stellt den Snapshot durch die Erstellung einer eindeutigen Kennung bereit. `VolumeGroupSnapshotContent` Objekt. Sie können Snapshots aus bestehenden PVCs erstellen und diese Snapshots als Datenquelle beim Erstellen neuer PVCs verwenden.

> ⓘ Der Lebenszyklus eines VolumeGroupSnapshot ist unabhängig vom Quell-PVC: Ein Snapshot bleibt auch dann erhalten, wenn der Quell-PVC gelöscht wird. Beim Löschen eines PVCs mit zugehörigen Snapshots markiert Trident das zugehörige Datenträgervolume im Status **Wird gelöscht**, entfernt es aber nicht vollständig. Der Snapshot der Volumengruppe wird entfernt, wenn alle zugehörigen Snapshots gelöscht werden.

Kubernetes `VolumeGroupSnapshotContent` Objekte

Ein Kubernetes `VolumeGroupSnapshotContent` Das Objekt stellt einen Gruppen-Snapshot dar, der von einem bereits bereitgestellten Volume erstellt wurde. Es ist analog zu einem `PersistentVolume` und kennzeichnet einen bereitgestellten Snapshot auf dem Speichercluster. Ähnlich `PersistentVolumeClaim` Und `PersistentVolume` Objekte, wenn ein Snapshot erstellt wird, `VolumeSnapshotContent` Das Objekt verwaltet eine Eins-zu-Eins-Zuordnung zum `VolumeSnapshot` Objekt, das die Erstellung des Snapshots angefordert hatte.

Der `VolumeGroupSnapshotContent` Das Objekt enthält Details, die die Snapshot-Gruppe identifizieren, wie zum Beispiel die `volumeGroupSnapshotHandle` und und einzelne VolumeSnapshotHandles, die auf dem Speichersystem vorhanden sind.

Wenn eine Snapshot-Anforderung eingeht, erstellt Trident im Backend den Volume-Group-Snapshot. Nachdem

der Snapshot der Volumengruppe erstellt wurde, konfiguriert Trident einen `VolumeGroupSnapshotContent` Das Objekt wird erstellt und somit der Kubernetes-API zugänglich gemacht.

Kubernetes `CustomResourceDefinition` Objekte

Kubernetes Custom Resources sind Endpunkte in der Kubernetes-API, die vom Administrator definiert werden und dazu dienen, ähnliche Objekte zu gruppieren. Kubernetes unterstützt die Erstellung benutzerdefinierter Ressourcen zum Speichern einer Sammlung von Objekten. Sie können diese Ressourcendefinitionen durch Ausführen von `kubectl get crds` .

Custom Resource Definitions (CRDs) und die zugehörigen Objektmetadaten werden von Kubernetes in seinem Metadatenspeicher abgelegt. Dadurch entfällt die Notwendigkeit eines separaten Ladengeschäfts für Trident.

Trident Anwendungen `CustomResourceDefinition` Objekte, um die Identität von Trident -Objekten wie Trident -Backends, Trident -Speicherklassen und Trident Volumes zu erhalten. Diese Objekte werden von Trident verwaltet. Darüber hinaus führt das CSI-Volume-Snapshot-Framework einige CRDs ein, die zur Definition von Volume-Snapshots erforderlich sind.

CRDs sind ein Konstrukt von Kubernetes. Objekte der oben definierten Ressourcen werden von Trident erstellt. Ein einfaches Beispiel: Wenn ein Backend erstellt wird mit `tridentctl` ein entsprechendes `tridentbackends` Ein CRD-Objekt wird zur Verwendung durch Kubernetes erstellt.

Hier einige Punkte, die Sie bei Tridents CRDs beachten sollten:

- Bei der Installation von Trident wird ein Satz von CRDs erstellt, die wie jeder andere Ressourcentyp verwendet werden können.
- Bei der Deinstallation von Trident mithilfe der `tridentctl uninstall` Der Befehl führt dazu, dass Trident -Pods gelöscht werden, die erstellten CRDs jedoch nicht bereinigt werden. Siehe"Trident deinstallieren" zu verstehen, wie Trident komplett entfernt und von Grund auf neu konfiguriert werden kann.

Trident `StorageClass` Objekte

Trident erstellt passende Speicherklassen für Kubernetes. `StorageClass` Objekte, die spezifizieren `csi.trident.netapp.io` in ihrem Bereitstellungsfeld. Der Name der Speicherklasse stimmt mit dem von Kubernetes überein. `StorageClass` Das Objekt, das es repräsentiert.

> Mit Kubernetes werden diese Objekte automatisch erstellt, wenn ein Kubernetes-Update durchgeführt wird. `StorageClass` Das System, das Trident als Provisionierer verwendet, ist registriert.

Speicherklassen umfassen eine Reihe von Anforderungen an Datenträger. Trident gleicht diese Anforderungen mit den Attributen jedes Speicherpools ab; wenn sie übereinstimmen, ist dieser Speicherpool ein gültiges Ziel für die Bereitstellung von Volumes mit dieser Speicherklasse.

Sie können Speicherklassenkonfigurationen erstellen, um Speicherklassen direkt über die REST-API zu definieren. Bei Kubernetes-Bereitstellungen gehen wir jedoch davon aus, dass sie bei der Registrierung neuer Kubernetes-Instanzen erstellt werden. `StorageClass` Objekte.

Trident Backend-Objekte

Backends stellen die Speicheranbieter dar, auf denen Trident Volumes bereitstellt; eine einzelne Trident

-Instanz kann beliebig viele Backends verwalten.

Dies ist einer der beiden Objekttypen, die Sie selbst erstellen und verwalten. Das andere ist Kubernetes. `StorageClass` Objekt.

Weitere Informationen zum Erstellen dieser Objekte finden Sie unter"Backends konfigurieren" .

Trident `StoragePool` Objekte

Speicherpools stellen die verschiedenen Speicherorte dar, die auf jedem Backend für die Bereitstellung zur Verfügung stehen. Für ONTAP entsprechen diese Aggregaten in SVMs. Bei NetApp HCI/ SolidFire entsprechen diese den vom Administrator festgelegten QoS-Bändern. Für den Cloud Volumes Service entsprechen diese den Regionen der Cloud-Anbieter. Jeder Speicherpool verfügt über eine Reihe unterschiedlicher Speicherattribute, die seine Leistungsmerkmale und Datenschutzeigenschaften definieren.

Im Gegensatz zu den anderen Objekten hier werden Speicherpoolkandidaten immer automatisch erkannt und verwaltet.

Trident `Volume` Objekte

Volumes sind die grundlegende Bereitstellungseinheit und umfassen Backend-Endpunkte wie NFS-Freigaben sowie iSCSI- und FC-LUNs. In Kubernetes entsprechen diese direkt `PersistentVolumes` . Wenn Sie ein Volume erstellen, stellen Sie sicher, dass es über eine Speicherklasse verfügt, die festlegt, wo dieses Volume bereitgestellt werden kann, sowie über eine Größe.

- In Kubernetes werden diese Objekte automatisch verwaltet. Sie können diese einsehen, um zu sehen, was Trident bereitgestellt hat.
- Beim Löschen eines PV mit zugehörigen Snapshots wird das entsprechende Trident Volume auf den Status **Wird gelöscht** aktualisiert. Damit das Trident -Volume gelöscht werden kann, müssen die Snapshots des Volumes entfernt werden.

Eine Volumenkonfiguration definiert die Eigenschaften, die ein bereitgestelltes Volumen haben soll.

Attribut	Typ	Erforderlich	Beschreibung
Version	Schnur	NEIN	Version der Trident API ("1")
Name	Schnur	Ja	Name des zu erstellenden Volumes
Speicherklasse	Schnur	Ja	Speicherklasse, die beim Bereitstellen des Volumes verwendet werden soll
Größe	Schnur	Ja	Größe des bereitzustellenden Datenvolumens in Bytes
Protokoll	Schnur	NEIN	Zu verwendender Protokolltyp: „Datei“ oder „Block“

Attribut	Typ	Erforderlich	Beschreibung
interner Name	Schnur	NEIN	Name des Objekts im Speichersystem; generiert von Trident
cloneSourceVolume	Schnur	NEIN	ontap (nas, san) & solidfire-*: Name des Volumes, von dem geklont werden soll
splitOnClone	Schnur	NEIN	ontap (nas, san): Trennt den Klon von seinem Elternknoten.
Snapshot-Richtlinie	Schnur	NEIN	ontap-*: Zu verwendende Snapshot-Richtlinie
Snapshot-Reserve	Schnur	NEIN	ontap-*: Prozentsatz des für Snapshots reservierten Volumens
Exportrichtlinie	Schnur	NEIN	ontap-nas*: Exportrichtlinie verwenden
Snapshot-Verzeichnis	bool	NEIN	ontap-nas*: Gibt an, ob das Snapshot-Verzeichnis sichtbar ist.
unixPermissions	Schnur	NEIN	ontap-nas*: Initial UNIX-Berechtigungen
Blockgröße	Schnur	NEIN	solidfire-*: Block-/Sektorgröße
Dateisystem	Schnur	NEIN	Dateisystemtyp

Trident erzeugt `internalName` beim Erstellen des Volumens. Dies besteht aus zwei Schritten. Zunächst wird das Speicherpräfix vorangestellt (entweder das Standardpräfix). `trident` oder dem Präfix in der Backend-Konfiguration) zum Volume-Namen, was zu einem Namen der Form führt `<prefix>-<volume-name>` . Anschließend wird der Name bereinigt, indem im Backend nicht zulässige Zeichen ersetzt werden. Bei ONTAP Backends werden Bindestriche durch Unterstriche ersetzt (dadurch wird der interne Name zu `<prefix>_<volume-name>`). Bei Element-Backends werden Unterstriche durch Bindestriche ersetzt.

Sie können Volume-Konfigurationen verwenden, um Volumes direkt über die REST-API bereitzustellen, aber bei Kubernetes-Bereitstellungen gehen wir davon aus, dass die meisten Benutzer die Standard-Kubernetes-Konfiguration verwenden. `PersistentVolumeClaim` Verfahren. Trident erstellt dieses Volume-Objekt automatisch im Rahmen des Bereitstellungsprozesses.

Trident `Snapshot` Objekte

Snapshots sind Momentaufnahmen von Datenträgern, die zur Bereitstellung neuer Datenträger oder zur Wiederherstellung des vorherigen Zustands verwendet werden können. In Kubernetes entsprechen diese direkt `VolumeSnapshotContent` Objekte. Jeder Snapshot ist mit einem Volume verknüpft, das die Datenquelle für den Snapshot darstellt.

Jede `Snapshot` Das Objekt umfasst die unten aufgeführten Eigenschaften:

Attribut	Typ	Erforderlich	Beschreibung
Version	Zeichenfolge	Ja	Version der Trident API ("1")
Name	Zeichenfolge	Ja	Name des Trident -Snapshot-Objekts
interner Name	Zeichenfolge	Ja	Name des Trident -Snapshot-Objekts auf dem Speichersystem
volumeName	Zeichenfolge	Ja	Name des persistenten Volumes, für das der Snapshot erstellt wurde
volumeInternalName	Zeichenfolge	Ja	Name des zugehörigen Trident Volume-Objekts im Speichersystem

In Kubernetes werden diese Objekte automatisch verwaltet. Sie können diese einsehen, um zu sehen, was Trident bereitgestellt hat.

Wenn ein Kubernetes `VolumeSnapshot` Wenn eine Objektanforderung erstellt wird, funktioniert Trident , indem ein Snapshot-Objekt auf dem zugrunde liegenden Speichersystem erstellt wird. Der `internalName` Dieses Snapshot-Objekt wird durch die Kombination des Präfixes generiert. `snapshot-` mit dem `UID` der `VolumeSnapshot` Objekt (zum Beispiel, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` Und `volumeInternalName` werden durch Abrufen der Details des zugrunde liegenden Volumens befüllt.

Trident `ResourceQuota` Objekt

Der Trident Dämonensatz verzehrt einen `system-node-critical` Prioritätsklasse – die höchste in Kubernetes verfügbare Prioritätsklasse – um sicherzustellen, dass Trident Volumes während des ordnungsgemäßen Herunterfahrens von Knoten identifizieren und bereinigen kann und dass Trident -Daemonset-Pods in Clustern mit hohem Ressourcendruck Workloads mit niedrigerer Priorität unterbrechen können.

Um dies zu erreichen, setzt Trident ein `ResourceQuota` Objekt, um sicherzustellen, dass eine Prioritätsklasse "system-node-critical" auf dem Trident -Daemonset erfüllt ist. Vor der Bereitstellung und der Erstellung des Daemonsets sucht Trident nach den `ResourceQuota` Objekt und, falls nicht gefunden, wendet es an.

Wenn Sie mehr Kontrolle über das Standardressourcenkontingent und die Prioritätsklasse benötigen, können Sie eine generieren. `custom.yaml` oder konfigurieren Sie die `ResourceQuota` Objekt, das das Helm-Chart verwendet.

Nachfolgend ein Beispiel für ein `ResourceQuota`-Objekt, das dem Trident -Daemonset Priorität einräumt.

```
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" .

Aufräumen `ResourceQuota` wenn die Installation fehlschlägt

In dem seltenen Fall, dass die Installation nach der `ResourceQuota` Objekt wurde erstellt, erster Versuch"Deinstallation" und dann neu installieren.

Sollte das nicht funktionieren, entfernen Sie die `ResourceQuota` Objekt.

Entfernen `ResourceQuota`

Wenn Sie Ihre Ressourcenzuweisung lieber selbst steuern möchten, können Sie den Trident entfernen. `ResourceQuota` Objekt mithilfe des Befehls:

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

Pod-Sicherheitsstandards (PSS) und Sicherheitskontextbeschränkungen (SCC)

Die Kubernetes Pod Security Standards (PSS) und Pod Security Policies (PSP) definieren Berechtigungsstufen und schränken das Verhalten von Pods ein. OpenShift Security Context Constraints (SCC) definieren analog dazu Pod-Beschränkungen, die spezifisch für die OpenShift Kubernetes Engine sind. Um diese Anpassung zu ermöglichen, aktiviert Trident während der Installation bestimmte Berechtigungen. In den folgenden Abschnitten werden die von Trident festgelegten Berechtigungen detailliert beschrieben.

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

Erforderlicher Kubernetes-Sicherheitskontext und zugehörige Felder

Erlaubnis	Beschreibung
Privilegiert	CSI erfordert bidirektionale Mount-Punkte, was bedeutet, dass auf dem Trident -Node-Pod ein privilegierter Container ausgeführt werden muss. Weitere Informationen finden Sie unter"Kubernetes: Mount-Propagation" .
Host-Netzwerk	Erforderlich für den iSCSI-Daemon. `iscsiadm` Verwaltet iSCSI-Mounts und nutzt das Host-Netzwerk zur Kommunikation mit dem iSCSI-Daemon.
Host-IPC	NFS nutzt Interprozesskommunikation (IPC) zur Kommunikation mit dem NFSD.
Host-PID	Erforderlich zum Starten `rpc-statd` für NFS. Trident fragt Hostprozesse ab, um festzustellen, ob `rpc-statd` wird vor dem Einbinden von NFS-Volumes ausgeführt.
Funktionen	Der `SYS_ADMIN` Diese Funktionalität wird als Teil der Standardfunktionen für privilegierte Container bereitgestellt. Docker legt beispielsweise diese Berechtigungen für privilegierte Container fest: `CapPrm: 0000003fffffffff` `CapEff: 0000003fffffffff`
Seccomp	Das Seccomp-Profil ist in privilegierten Containern immer "Unconfided"; daher kann es in Trident nicht aktiviert werden.
SELinux	Auf OpenShift werden privilegierte Container in der `spc_t` ("Super Privileged Container")-Domäne, und unprivilegierte Container werden in der `container_t` Domain. An `containerd` , mit `container-selinux` installiert, alle Container werden in der `spc_t` Domain, wodurch SELinux effektiv deaktiviert wird. Daher fügt Trident nichts hinzu `seLinuxOptions` zu Containern.
DAC	Privilegierte Container müssen als Root ausgeführt werden. Nicht privilegierte Container werden als Root ausgeführt, um auf die von CSI benötigten Unix-Sockets zuzugreifen.

Pod-Sicherheitsstandards (PSS)

Etikett	Beschreibung	Standard
`pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version`	Ermöglicht es dem Trident Controller und den Knoten, in den Installations-Namespace aufgenommen zu werden. Ändern Sie die Namespace-Bezeichnung nicht.	`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 eingeplant werden, eine Fehlermeldung wie „Fehler beim Erstellen: …“ oder „Warnung: trident-csi-…“ erscheint. Wenn dies der Fall ist, prüfen Sie, ob die Namespace-Bezeichnung für `privileged` wurde geändert. Installieren Sie Trident in diesem Fall neu.

Pod-Sicherheitsrichtlinien (PSP)

Feld	Beschreibung	Standard
`allowPrivilegeEscalation`	Privilegierte Container müssen eine Rechteausweitung ermöglichen.	`true`
`allowedCSIDrivers`	Trident verwendet keine Inline-CSI-Ephemeralvolumina.	Leer
`allowedCapabilities`	Nicht privilegierte Trident Container benötigen keine weiteren Berechtigungen als die Standardberechtigungen, während privilegierten Containern alle möglichen Berechtigungen gewährt werden.	Leer
`allowedFlexVolumes`	Trident verwendet keine"FlexVolume-Treiber" Daher sind sie nicht in der Liste der zulässigen Bände enthalten.	Leer
`allowedHostPaths`	Der Trident -Node-Pod mountet das Root-Dateisystem des Nodes, daher bringt das Festlegen dieser Liste keinen Vorteil.	Leer
`allowedProcMountTypes`	Trident verwendet keine `ProcMountTypes` .	Leer
`allowedUnsafeSysctls`	Trident benötigt keine unsicheren… `sysctls` .	Leer
`defaultAddCapabilities`	Für privilegierte Container müssen keine zusätzlichen Funktionen hinzugefügt werden.	Leer
`defaultAllowPrivilegeEscalation`	Die Gewährung von Privilegienerweiterungen wird in jedem Trident -Pod gehandhabt.	`false`
`forbiddenSysctls`	NEIN `sysctls` sind erlaubt.	Leer

Feld	Beschreibung	Standard
`fsGroup`	Trident Container werden als Root ausgeführt.	`RunAsAny`
`hostIPC`	Das Einbinden von NFS-Volumes erfordert die Kommunikation zwischen Host und IPC. `nfsd`	`true`
`hostNetwork`	iscsiadm benötigt das Host-Netzwerk zur Kommunikation mit dem iSCSI-Daemon.	`true`
`hostPID`	Die Host-PID wird benötigt, um zu prüfen, ob `rpc-statd` läuft auf dem Knoten.	`true`
`hostPorts`	Trident verwendet keine Host-Ports.	Leer
`privileged`	Trident Node-Pods müssen einen privilegierten Container ausführen, um Volumes einzubinden.	`true`
`readOnlyRootFilesystem`	Trident Node-Pods müssen in das Node-Dateisystem schreiben.	`false`
`requiredDropCapabilities`	Trident -Node-Pods führen einen privilegierten Container aus und können keine Capabilities verlieren.	`none`
`runAsGroup`	Trident Container werden als Root ausgeführt.	`RunAsAny`
`runAsUser`	Trident Container werden als Root ausgeführt.	`runAsAny`
`runtimeClass`	Trident verwendet nicht `RuntimeClasses` .	Leer
`seLinux`	Trident lässt sich nicht einstellen `seLinuxOptions` weil es derzeit Unterschiede in der Art und Weise gibt, wie Container-Laufzeitumgebungen und Kubernetes-Distributionen SELinux handhaben.	Leer
`supplementalGroups`	Trident Container werden als Root ausgeführt.	`RunAsAny`
`volumes`	Trident Pods benötigen diese Volumen-Plugins.	`hostPath, projected, emptyDir`

Sicherheitskontextbeschränkungen (SCC)

Labels	Beschreibung	Standard
`allowHostDirVolumePlugin`	Trident Node-Pods mounten das Root-Dateisystem des Nodes.	`true`
`allowHostIPC`	Das Einbinden von NFS-Volumes erfordert die Kommunikation zwischen Host und IPC. `nfsd`.	`true`
`allowHostNetwork`	iscsiadm benötigt das Host-Netzwerk zur Kommunikation mit dem iSCSI-Daemon.	`true`
`allowHostPID`	Die Host-PID wird benötigt, um zu prüfen, ob `rpc-statd` läuft auf dem Knoten.	`true`
`allowHostPorts`	Trident verwendet keine Host-Ports.	`false`
`allowPrivilegeEscalation`	Privilegierte Container müssen eine Rechteausweitung ermöglichen.	`true`
`allowPrivilegedContainer`	Trident Node-Pods müssen einen privilegierten Container ausführen, um Volumes einzubinden.	`true`
`allowedUnsafeSysctls`	Trident benötigt keine unsicheren… `sysctls`.	`none`
`allowedCapabilities`	Nicht privilegierte Trident Container benötigen keine weiteren Berechtigungen als die Standardberechtigungen, während privilegierten Containern alle möglichen Berechtigungen gewährt werden.	Leer
`defaultAddCapabilities`	Für privilegierte Container müssen keine zusätzlichen Funktionen hinzugefügt werden.	Leer
`fsGroup`	Trident Container werden als Root ausgeführt.	`RunAsAny`
`groups`	Diese SCC ist spezifisch für Trident und an ihren Benutzer gebunden.	Leer
`readOnlyRootFilesystem`	Trident Node-Pods müssen in das Node-Dateisystem schreiben.	`false`
`requiredDropCapabilities`	Trident -Node-Pods führen einen privilegierten Container aus und können keine Capabilities verlieren.	`none`
`runAsUser`	Trident Container werden als Root ausgeführt.	`RunAsAny`

Labels	Beschreibung	Standard
`seLinuxContext`	Trident lässt sich nicht einstellen `seLinuxOptions` weil es derzeit Unterschiede in der Art und Weise gibt, wie Container-Laufzeitumgebungen und Kubernetes-Distributionen SELinux handhaben.	Leer
`seccompProfiles`	Privilegierte Container laufen immer "Unconfided".	Leer
`supplementalGroups`	Trident Container werden als Root ausgeführt.	`RunAsAny`
`users`	Es wird ein Eintrag bereitgestellt, um diese SCC an den Trident Benutzer im Trident -Namensraum zu binden.	n/a
`volumes`	Trident Pods benötigen diese Volumen-Plugins.	`hostPath, downwardAPI, projected, emptyDir`