Riferimento

Trident

NetApp
July 01, 2026

Sommario

Riferimento

Porte Trident

Scopri di più sulle porte che Trident utilizza per la comunicazione.

Panoramica

Trident utilizza diverse porte per la comunicazione all'interno dei cluster Kubernetes e con i backend di storage. Di seguito è riportato un riepilogo delle porte principali, delle loro funzioni e delle considerazioni sulla sicurezza.

- **Outbound focus**: i nodi Kubernetes (controller e worker) avviano principalmente il traffico verso LIF/IP di storage, pertanto le regole di iptables dovrebbero consentire outbound dagli IP dei nodi verso IP di storage specifici su queste porte. Evitare regole generiche "any-to-any".
- **Restrizioni in ingresso**: limita le porte Trident interne al traffico interno al cluster (ad esempio, utilizzando CNI come Calico). Nessuna esposizione in ingresso non necessaria sui firewall host.
- **Sicurezza del protocollo**:
  - Utilizzare TCP ove possibile (più affidabile).
  - Abilita CHAP/IPsec per iSCSI se sensibile; TLS/HTTPS per la gestione (porta 443/8443).
  - Per NFSv4 (predefinito in Trident), elimina le porte UDP/NFSv3 più vecchie (ad esempio, 4045-4049) se non necessarie.
  - Limitare alle subnet attendibili; monitorare con strumenti come Prometheus (porta opzionale 8001).

Porte per i nodi controller

Queste porte sono principalmente destinate all'operatore Trident (gestione backend). Tutte le porte interne sono a livello di pod; consentirle sui nodi solo se il firewall host interferisce con CNI.

Porta/Protocollo	Direzione	Scopo	Driver/Protocollo	Note di sicurezza
TCP 8000	Inbound/Outbound (interno al cluster)	Trident REST server (comunicazioni operator-controller)	Tutti	Limitare ai CIDR del pod; nessuna esposizione esterna.
TCP 8443	Inbound/Outbound (interno al cluster)	Backchannel HTTPS (API interna sicura)	Tutti	Crittografato tramite TLS; limitare al service mesh Kubernetes se utilizzato.
TCP 8001	Inbound (interno al cluster, facoltativo)	metriche Prometheus	Tutti	Esporre solo agli strumenti di monitoraggio (ad esempio, utilizzando RBAC); disabilitare se inutilizzato.
TCP 443	outbound	HTTPS a ONTAP SVM/cluster mgmt LIF	ONTAP (tutti), ANF	Richiedi la convalida del certificato TLS; limita solo agli IP LIF di gestione.

Porta/Protocollo	Direzione	Scopo	Driver/Protocollo	Note di sicurezza
TCP 8443	outbound	HTTPS al proxy dei servizi Web E-Series	E-Series (iSCSI)	API REST predefinita; usa certs; configurabile nel backend YAML.

Porte per i nodi worker

Queste porte sono destinate ai daemonset dei nodi CSI e ai mount dei pod. Le porte dati sono outbound verso i LIF dei dati del sistema storage; includere le porte extra NFSv3 se si utilizza NFSv3 (opzionale per NFSv4).

Porta/Protocollo	Direzione	Scopo	Driver/Protocollo	Note di sicurezza
TCP 17546	In entrata (locale al pod)	Sonde di liveness/readiness del nodo CSI	Tutti	Configurabile (--probe-port); assicurarsi che non ci siano conflitti host; solo locale.
TCP 8000	Inbound/Outbound (interno al cluster)	Server REST Trident	Tutti	Come sopra; interno al pod.
TCP 8443	Inbound/Outbound (interno al cluster)	Backchannel HTTPS	Tutti	Come sopra.
TCP 8001	Inbound (interno al cluster, facoltativo)	metriche Prometheus	Tutti	Come sopra.
TCP 443	outbound	HTTPS a ONTAP SVM/cluster mgmt LIF	ONTAP (tutti), ANF	Come sopra; utilizzato per la discovery.
TCP 8443	outbound	HTTPS al proxy dei servizi Web E-Series	E-Series (iSCSI)	Come sopra.
TCP/UDP 111	outbound	RPCBIND/portmapper	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Obbligatorio per v3; facoltativo per v4 (firewall offload); limitare se si utilizza solo NFSv4.
TCP/UDP 2049	outbound	Demone NFS	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Dati principali; ben noti; utilizzare TCP per affidabilità.
TCP/UDP 635	outbound	Daemon di mount	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Montaggio; possibili callback bidirezionali (consentire inbound effimero se necessario).
UDP 4045	outbound	NFS lock manager (nlockmgr)	ONTAP-NAS (NFSv3)	Blocco dei file; saltare per v4 (pNFS gestisce); solo UDP.

Porta/Protocollo	Direzione	Scopo	Driver/Protocollo	Note di sicurezza
UDP 4046	outbound	Monitor dello stato NFS (statd)	ONTAP-NAS (NFSv3)	Notifiche; potrebbero essere necessarie porte effimere inbound (1024-65535) per i callback.
UDP 4049	outbound	Demone quota NFS (rquotad)	ONTAP-NAS (NFSv3)	Quote; saltare per v4.
TCP 3260	outbound	iSCSI target (rilevamento/dati/CHAP)	ONTAP-SAN (iSCSI), E-Series (iSCSI)	Ben noto; autenticazione CHAP su questa porta; abilita mutual CHAP per la sicurezza.
TCP 445	outbound	SMB/CIFS	ONTAP-NAS (SMB), ANF (SMB)	Ben noto; utilizzare SMB3 con crittografia (Trident annotation netapp.io/smb-encryption=true).
TCP/UDP 88 (facoltativo)	outbound	Autenticazione Kerberos	ONTAP (NFS/SMB/iSCSI con Kerb)	Se si utilizza Kerberos (non predefinito); ai server AD, non al sistema storage.
TCP/UDP 389 (facoltativo)	outbound	LDAP	ONTAP (NFS/SMB con LDAP)	Simile; per la risoluzione dei nomi/autenticazione; limitare ad AD.

La porta di liveness/readiness probe può essere modificata durante l'installazione utilizzando il `--probe-port` flag. È importante assicurarsi che questa porta non sia utilizzata da un altro processo sui nodi worker.

API REST Trident

Sebbene "comandi e opzioni di tridentctl" siano il modo più semplice per interagire con la Trident REST API, puoi utilizzare direttamente l'endpoint REST se lo preferisci.

Quando utilizzare l'API REST

L'API REST è utile per installazioni avanzate che utilizzano Trident come binario autonomo in distribuzioni non-Kubernetes.

Per una maggiore sicurezza, Trident `REST API` è limitato per impostazione predefinita a localhost quando viene eseguito all'interno di un pod. Per modificare questo comportamento, è necessario impostare l'argomento di Trident `-address` nella configurazione del pod.

Utilizzo delle REST API

Per esempi di come vengono chiamate queste API, passa il flag di debug (`-d`. Per ulteriori informazioni, fai riferimento a "Gestisci Trident usando tridentctl".

L'API funziona come segue:

GET

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

Elenca tutti gli oggetti di quel tipo.

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

Ottiene i dettagli dell'oggetto denominato.

POST

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

Crea un oggetto del tipo specificato.

- Richiede una configurazione JSON per l'oggetto da creare. Per le specifiche di ciascun tipo di oggetto, fare riferimento a "Gestisci Trident usando tridentctl".
- Se l'oggetto esiste già, il comportamento varia: i backend aggiornano l'oggetto esistente, mentre tutti gli altri tipi di oggetto non riusciranno nell'operazione.

ELIMINA

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

Elimina la risorsa denominata.

> I volumi associati ai backend o alle classi di archiviazione continueranno a esistere; questi devono essere eliminati separatamente. Per ulteriori informazioni, fai riferimento a "Gestisci Trident usando tridentctl".

Opzioni della riga di comando

Trident espone diverse opzioni da riga di comando per l'orchestratore Trident. È possibile utilizzare queste opzioni per modificare la distribuzione.

Registrazione

`-debug`

Abilita l'output di debug.

`-loglevel <level>`

Imposta il livello di registrazione (debug, info, warn, error, fatal). Il valore predefinito è info.

Kubernetes

`-k8s_pod`

Utilizzare questa opzione o `-k8s_api_server` per abilitare il supporto Kubernetes. Impostando questa opzione, Trident utilizza le credenziali dell'account di servizio Kubernetes del pod contenente per contattare il server API. Questa funzione funziona solo quando Trident viene eseguito come pod in un cluster Kubernetes con account di servizio abilitati.

-k8s_api_server <insecure-address:insecure-port>

Utilizzare questa opzione o `-k8s_pod` per abilitare il supporto Kubernetes. Quando specificato, Trident si connette al server API Kubernetes utilizzando l'indirizzo e la porta non sicuri forniti. Questo consente a Trident di essere distribuito al di fuori di un pod; tuttavia, supporta solo connessioni non sicure al server API. Per connettersi in modo sicuro, distribuire Trident in un pod con l' `-k8s_pod` opzione.

Docker

-volume_driver <name>

Nome del driver utilizzato durante la registrazione del plugin Docker. Il valore predefinito è `netapp`.

-driver_port <port-number>

Ascolta su questa porta anziché su un socket di dominio UNIX.

-config <file>

Obbligatorio; è necessario specificare questo percorso per un file di configurazione backend.

REST

-address <ip-or-host>

Specifica l'indirizzo su cui il server REST di Trident deve essere in ascolto. Il valore predefinito è localhost. Quando si è in ascolto su localhost e si esegue all'interno di un pod Kubernetes, l'interfaccia REST non è direttamente accessibile dall'esterno del pod. Utilizzare `-address ""` per rendere l'interfaccia REST accessibile dall'indirizzo IP del pod.

L'interfaccia REST di Trident può essere configurata per ascoltare e servire solo su 127.0.0.1 (per IPv4) o [::1] (per IPv6).

-port <port-number>

Specifica la porta su cui il server REST di Trident deve essere in ascolto. Il valore predefinito è 8000.

-rest

Abilita l'interfaccia REST. Il valore predefinito è true.

Oggetti Kubernetes e Trident

È possibile interagire con Kubernetes e Trident utilizzando le API REST leggendo e scrivendo oggetti risorsa. Esistono diversi oggetti risorsa che determinano la relazione tra Kubernetes e Trident, Trident e storage, e Kubernetes e storage. Alcuni di questi oggetti sono gestiti tramite Kubernetes e gli altri sono gestiti tramite Trident.

Come interagiscono gli oggetti tra loro?

Forse il modo più semplice per capire gli oggetti, a cosa servono e come interagiscono, è seguire una singola richiesta di storage da parte di un utente Kubernetes:

1. Un utente crea un `PersistentVolumeClaim` richiedendo un nuovo `PersistentVolume` di una particolare dimensione da un `StorageClass` Kubernetes precedentemente configurato dall'amministratore.

2. Kubernetes `StorageClass` identifica Trident come provisioner e include parametri che indicano a Trident come effettuare il provisioning di un volume per la classe richiesta.
3. Trident esamina il proprio `StorageClass` con lo stesso nome che identifica i corrispondenti `Backends` e `StoragePools` che può utilizzare per effettuare il provisioning dei volumi per la classe.
4. Trident fornisce storage su un backend corrispondente e crea due oggetti: un `PersistentVolume` in Kubernetes che indica a Kubernetes come trovare, montare e trattare il volume, e un volume in Trident che mantiene la relazione tra `PersistentVolume` e lo storage effettivo.
5. Kubernetes lega `PersistentVolumeClaim` al nuovo `PersistentVolume`. I pod che includono `PersistentVolumeClaim` montano quel PersistentVolume su qualsiasi host su cui vengono eseguiti.
6. Un utente crea un `VolumeSnapshot` di un PVC esistente, utilizzando un `VolumeSnapshotClass` che punta a Trident.
7. Trident identifica il volume associato al PVC e crea una snapshot del volume sul suo backend. Crea anche un `VolumeSnapshotContent` che istruisce Kubernetes su come identificare la snapshot.
8. Un utente può creare un `PersistentVolumeClaim` usando `VolumeSnapshot` come sorgente.
9. Trident identifica la Snapshot richiesta ed esegue la stessa serie di passaggi necessari per creare una `PersistentVolume` e una `Volume`.

Per ulteriori informazioni sugli oggetti Kubernetes, si consiglia vivamente di leggere la sezione "Volumi persistenti" della documentazione Kubernetes.

Oggetti `PersistentVolumeClaim` Kubernetes

Un oggetto Kubernetes `PersistentVolumeClaim` è una richiesta di storage effettuata da un utente del cluster Kubernetes.

Oltre alle specifiche standard, Trident consente agli utenti di specificare le seguenti annotazioni specifiche per i volumi se desiderano sovrascrivere i valori predefiniti che hai impostato nella configurazione del backend:

Annotazione	Opzione volume	Driver supportati
trident.netapp.io/fileSystem	fileSystem	ontap-san, solidfire-san,ontap-san-economy
trident.netapp.io/cloneFromPVC	cloneSourceVolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ontap-nas, ontap-san
trident.netapp.io/protocol	protocollo	qualsiasi
trident.netapp.io/exportPolicy	exportPolicy	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
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
trident.netapp.io/snapshotDirectory	snapshotDirectory	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup

Annotazione	Opzione volume	Driver supportati
trident.netapp.io/unixPermissions	unixPermissions	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/blockSize	blockSize	solidfire-san
trident.netapp.io/skipRecoveryQueue	skipRecoveryQueue	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy

Se il PV creato ha la `Delete` reclaim policy, Trident elimina sia il PV che il backing volume quando il PV viene rilasciato (cioè quando l'utente elimina il PVC). Se l'azione di eliminazione non riesce, Trident contrassegna il PV come tale e ritenta periodicamente l'operazione finché non ha successo o il PV viene eliminato manualmente. Se il PV utilizza la `Retain` policy, Trident lo ignora e presume che l'amministratore lo rimuova da Kubernetes e dal backend, consentendo il backup o l'ispezione del volume prima della sua rimozione. Si noti che l'eliminazione del PV non causa a Trident di eliminare il backing volume. Dovresti rimuoverlo utilizzando la REST API (`tridentctl`).

Trident supporta la creazione di Volume Snapshot utilizzando la specifica CSI: è possibile creare una Volume Snapshot e utilizzarla come Data Source per clonare i PVC esistenti. In questo modo, copie point-in-time dei PV possono essere esposte a Kubernetes sotto forma di snapshot. Le snapshot possono quindi essere utilizzate per creare nuovi PV. Dai un'occhiata a `On-Demand Volume Snapshots` per vedere come funzionerebbe.

Trident fornisce anche le `cloneFromPVC` e `splitOnClone` annotazioni per la creazione di cloni. Puoi utilizzare queste annotazioni per clonare un PVC senza dover utilizzare l'implementazione CSI.

Ecco un esempio: se un utente ha già un PVC chiamato `mysql`, l'utente può creare un nuovo PVC chiamato `mysqlclone` utilizzando l'annotazione, come `trident.netapp.io/cloneFromPVC: mysql`. Con questa annotazione impostata, Trident clona il volume corrispondente al PVC mysql, invece di eseguire il provisioning di un volume da zero.

Considera i seguenti punti:

- NetApp consiglia di clonare un volume inattivo.
- Un PVC e il suo clone devono trovarsi nello stesso namespace Kubernetes e avere la stessa storage class.
- Con i driver `ontap-nas` e `ontap-san`, potrebbe essere auspicabile impostare l'annotazione PVC `trident.netapp.io/splitOnClone` in combinazione con `trident.netapp.io/cloneFromPVC`. Con `trident.netapp.io/splitOnClone` impostato su `true`, Trident separa il volume clonato dal volume d'origine e quindi disaccoppia completamente il ciclo di vita del volume clonato dal suo volume d'origine, a scapito di perdere una certa efficienza di storage. Non impostare `trident.netapp.io/splitOnClone` o impostarlo su `false` comporta una riduzione del consumo di spazio sul backend, a scapito della creazione di dipendenze tra il volume d'origine e il volume clone, tali per cui il volume d'origine non può essere eliminato a meno che il clone non venga eliminato prima. Uno scenario in cui ha senso separare il clone è la clonazione di un volume database vuoto, dove ci si aspetta che il volume e il suo clone divergano notevolmente e non beneficino delle efficienze di storage offerte da ONTAP.

La `sample-input` directory contiene esempi di definizioni di PVC da utilizzare con Trident. Fare riferimento a per una descrizione completa dei parametri e delle impostazioni associate ai volumi Trident.

Oggetti `PersistentVolume` Kubernetes

Un oggetto Kubernetes `PersistentVolume` rappresenta un pezzo di storage messo a disposizione del cluster Kubernetes. Ha un ciclo di vita indipendente dal pod che lo utilizza.

> Trident crea `PersistentVolume` oggetti e li registra automaticamente con il cluster Kubernetes in base ai volumi che fornisce. Non è necessario gestirli personalmente.

Quando si crea un PVC che fa riferimento a un sistema basato su Trident `StorageClass`, Trident esegue il provisioning di un nuovo volume utilizzando la corrispondente storage class e registra un nuovo PV per quel volume. Nel configurare il volume fornito e il PV corrispondente, Trident segue le seguenti regole:

- Trident genera un nome PV per Kubernetes e un nome interno che utilizza per il provisioning dello storage. In entrambi i casi, si assicura che i nomi siano unici nel loro ambito.
- La dimensione del volume corrisponde il più possibile a quella richiesta nel PVC, anche se potrebbe essere arrotondata alla quantità allocabile più vicina, a seconda della piattaforma.

Oggetti `StorageClass` Kubernetes

Gli oggetti Kubernetes `StorageClass` sono specificati per nome in `PersistentVolumeClaims` per fornire storage con un insieme di proprietà. La storage class stessa identifica il provisioner da utilizzare e definisce quell'insieme di proprietà in termini che il provisioner comprende.

È uno dei due oggetti di base che devono essere creati e gestiti dall'amministratore. L'altro è l'oggetto backend Trident.

Un oggetto Kubernetes `StorageClass` che utilizza Trident appare così:

```
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
```

Questi parametri sono specifici di Trident e indicano a Trident come effettuare il provisioning dei volumi per la classe.

I parametri della storage class sono:

Attributo	Tipo	Richiesto	Descrizione
attributi	map[string]string	no	Vedere la sezione attributi qui sotto

Attributo	Tipo	Richiesto	Descrizione
storagePools	map[string]StringList	no	Mappa dei nomi dei backend agli elenchi di pool di storage all'interno
additionalStoragePools	map[string]StringList	no	Mappa dei nomi dei backend agli elenchi di pool di storage all'interno
excludeStoragePools	map[string]StringList	no	Mappa dei nomi dei backend agli elenchi di pool di storage all'interno

Gli attributi di storage e i loro possibili valori possono essere classificati in attributi di selezione del pool di storage e attributi di Kubernetes.

Attributi di selezione del pool di storage

Questi parametri determinano quali pool di storage gestiti da Trident devono essere utilizzati per effettuare il provisioning dei volumi di un determinato tipo.

Attributo	Tipo	Valori	Offerta	Richiesta	Supportato da
media[1]	stringa	hdd, hybrid, ssd	Il pool contiene supporti di questo tipo; ibrido significa entrambi	Tipo di media specificato	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
provisioningType	stringa	sottile, spesso	Il pool supporta questo metodo di provisioning	Metodo di provisioning specificato	spesso: tutti ontap; sottile: tutti ontap & solidfire-san
backendType	stringa	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	Il pool appartiene a questo tipo di backend	Backend specificato	Tutti i driver
istantanee	bool	vero, falso	Il pool supporta volumi con snapshot	Volume con snapshot abilitato	ontap-nas, ontap-san, solidfire-san
cloni	bool	vero, falso	Il pool supporta la clonazione dei volumi	Volume con cloni abilitati	ontap-nas, ontap-san, solidfire-san

Attributo	Tipo	Valori	Offerta	Richiesta	Supportato da
crittografia	bool	vero, falso	Il pool supporta volumi criptati	Volume con crittografia abilitata	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	int	intero positivo	Il pool è in grado di garantire IOPS in questo intervallo	Volume garantisce questi IOPS	solidfire-san

[1]: Non supportato dai sistemi ONTAP Select

Nella maggior parte dei casi, i valori richiesti influenzano direttamente il provisioning; ad esempio, la richiesta di thick provisioning si traduce in un volume thickly provisioned. Tuttavia, un pool di storage Element utilizza il minimo e il massimo IOPS offerti per impostare i valori QoS, piuttosto che il valore richiesto. In questo caso, il valore richiesto viene utilizzato solo per selezionare il pool di storage.

Idealmente, puoi usare `attributes` da solo per modellare le qualità dello storage di cui hai bisogno per soddisfare le esigenze di una particolare classe. Trident scopre e seleziona automaticamente i pool di storage che corrispondono a *tutti* i `attributes` che specifichi.

Se non riesci a usare `attributes` per selezionare automaticamente i pool giusti per una classe, puoi usare i parametri `storagePools` e `additionalStoragePools` per affinare ulteriormente i pool o anche per selezionare un insieme specifico di pool.

È possibile utilizzare il parametro `storagePools` per restringere ulteriormente l'insieme dei pool che corrispondono a qualsiasi `attributes` specificato. In altre parole, Trident utilizza l'intersezione dei pool identificati dai parametri `attributes` e `storagePools` per il provisioning. È possibile utilizzare uno dei due parametri da solo o entrambi insieme.

È possibile utilizzare il parametro `additionalStoragePools` per estendere l'insieme di pool che Trident utilizza per il provisioning, indipendentemente da eventuali pool selezionati dai `attributes` e `storagePools` parametri.

È possibile utilizzare il parametro `excludeStoragePools` per filtrare l'insieme dei pool che Trident utilizza per il provisioning. L'utilizzo di questo parametro rimuove tutti i pool che corrispondono.

Nei `storagePools` e `additionalStoragePools` parametri, ogni voce assume la forma `<backend>:<storagePoolList>`, dove `<storagePoolList>` è un elenco separato da virgole di storage pool per il backend specificato. Ad esempio, un valore per `additionalStoragePools` potrebbe essere simile a `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`. Questi elenchi accettano valori regex sia per il backend che per i valori dell'elenco. Puoi usare `tridentctl get backend` per ottenere l'elenco dei backend e dei loro pool.

Attributi di Kubernetes

Questi attributi non hanno alcun impatto sulla selezione di pool di storage/backend da parte di Trident durante il provisioning dinamico. Invece, questi attributi forniscono semplicemente parametri supportati da Kubernetes Persistent Volumes. I nodi worker sono responsabili delle operazioni di creazione del file system e potrebbero richiedere utility di file system, come xfsprogs.

Attributo	Tipo	Valori	Descrizione	Driver rilevanti	Versione Kubernetes
fsType	stringa	ext4, ext3, xfs	Il tipo di file system per i volumi a blocchi	solidfire-san, ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy	Tutti
allowVolumeExpansion	booleano	vero, falso	Abilitare o disabilitare il supporto per aumentare le dimensioni del PVC	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, azure-netapp-files	1.11+
volumeBindingMode	stringa	Immediato, WaitForFirstConsumer	Scegli quando avviene il binding dei volumi e il provisioning dinamico	Tutti	1.19 - 1.26

- Il `fsType` parametro viene utilizzato per controllare il tipo di file system desiderato per le SAN LUN. Inoltre, Kubernetes utilizza anche la presenza di `fsType` in una storage class per indicare che esiste un file system. La proprietà del volume può essere controllata utilizzando il `fsGroup` security context di un pod solo se `fsType` è impostato. Fare riferimento a "Kubernetes: Configurare un Security Context per un pod o un container" per una panoramica sull'impostazione della proprietà del volume utilizzando il `fsGroup` context. Kubernetes applicherà il valore `fsGroup` solo se:

  - `fsType` è impostato nella storage class.
  - La modalità di accesso del PVC è RWO.

  Per i driver di storage NFS, un file system esiste già come parte dell'esportazione NFS. Per utilizzare `fsGroup` la storage class deve comunque specificare un `fsType`. Puoi impostarlo su `nfs` o su qualsiasi valore non nullo.

- Consultare "Espandi volumi" per ulteriori dettagli sull'espansione del volume.
- Il bundle di installazione di Trident fornisce diversi esempi di definizione di storage class da utilizzare con Trident in `sample-input/storage-class-*.yaml`. L'eliminazione di una storage class Kubernetes causa anche l'eliminazione della corrispondente storage class Trident.

Oggetti `VolumeSnapshotClass` Kubernetes

Gli oggetti di Kubernetes `VolumeSnapshotClass` sono analoghi a `StorageClasses`. Aiutano a definire più

classi di storage e sono referenziati dalle istantanee di volume per associare l'istantanea alla classe di istantanea richiesta. Ogni istantanea di volume è associata a una singola classe di istantanea di volume.

Un `VolumeSnapshotClass` dovrebbe essere definito da un amministratore per creare le istantanee. Una classe di istantanee di volume viene creata con la seguente definizione:

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

Il `driver` specifica a Kubernetes che le richieste di snapshot di volume della classe `csi-snapclass` sono gestite da Trident. Il `deletionPolicy` specifica l'azione da intraprendere quando uno snapshot deve essere eliminato. Quando `deletionPolicy` è impostato su `Delete`, gli oggetti snapshot del volume così come lo snapshot sottostante sul cluster di storage vengono rimossi quando uno snapshot viene eliminato. In alternativa, impostandolo su `Retain` significa che `VolumeSnapshotContent` e lo snapshot fisico vengono mantenuti.

Oggetti `VolumeSnapshot` Kubernetes

Un oggetto Kubernetes `VolumeSnapshot` è una richiesta di creazione di una snapshot di un volume. Così come un PVC rappresenta una richiesta fatta da un utente per un volume, una snapshot di volume è una richiesta fatta da un utente per creare una snapshot di un PVC esistente.

Quando arriva una richiesta di snapshot del volume, Trident gestisce automaticamente la creazione dello snapshot per il volume sul backend ed espone lo snapshot creando un oggetto `VolumeSnapshotContent` univoco. Puoi creare snapshot da PVC esistenti e utilizzare gli snapshot come DataSource quando crei nuovi PVC.

> Il ciclo di vita di un VolumeSnapshot è indipendente dal PVC di origine: una snapshot persiste anche dopo che il PVC di origine è stato eliminato. Quando si elimina un PVC che ha snapshot associate, Trident contrassegna il volume di supporto per questo PVC in stato **Deleting**, ma non lo rimuove completamente. Il volume viene rimosso quando tutte le snapshot associate vengono eliminate.

Oggetti `VolumeSnapshotContent` Kubernetes

Un oggetto Kubernetes `VolumeSnapshotContent` rappresenta una snapshot presa da un volume già fornito. È analogo a `PersistentVolume` e indica una snapshot fornita sul cluster di storage. Analogamente a `PersistentVolumeClaim` e `PersistentVolume` oggetti, quando viene creata una snapshot, l'oggetto `VolumeSnapshotContent` mantiene una mappatura uno-a-uno con l'oggetto `VolumeSnapshot` che ha richiesto la creazione della snapshot.

L'`VolumeSnapshotContent`oggetto contiene dettagli che identificano in modo univoco la snapshot, come il `snapshotHandle`. Questo `snapshotHandle` è una combinazione univoca del nome del PV e del nome dell' `VolumeSnapshotContent`oggetto.

Quando arriva una richiesta di snapshot, Trident crea lo snapshot sul backend. Dopo che lo snapshot è stato

creato, Trident configura un `VolumeSnapshotContent` oggetto e quindi espone lo snapshot all'API di Kubernetes.

> In genere, non è necessario gestire l'oggetto `VolumeSnapshotContent`. Fa eccezione il caso in cui si voglia "importare una snapshot del volume" creato al di fuori di Trident.

Oggetti `VolumeGroupSnapshotClass` Kubernetes

Gli oggetti di Kubernetes `VolumeGroupSnapshotClass` sono analoghi a `VolumeSnapshotClass`. Aiutano a definire più classi di storage e sono referenziati dalle snapshot del gruppo di volumi per associare la snapshot alla classe di snapshot richiesta. Ogni snapshot del gruppo di volumi è associata a una singola classe di snapshot del gruppo di volumi.

A `VolumeGroupSnapshotClass` deve essere definito da un amministratore per creare un gruppo di snapshot. Una classe di snapshot di gruppo di volumi viene creata con la seguente definizione:

```
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
```

Il `driver` specifica a Kubernetes che le richieste di snapshot di gruppo di volumi della `csi-group-snap-class` classe sono gestite da Trident. Il `deletionPolicy` specifica l'azione da intraprendere quando una snapshot di gruppo deve essere eliminata. Quando `deletionPolicy` è impostato su `Delete`, gli oggetti snapshot del gruppo di volumi e lo snapshot sottostante sul cluster di storage vengono rimossi quando uno snapshot viene eliminato. In alternativa, impostandolo su `Retain` significa che `VolumeGroupSnapshotContent` e lo snapshot fisico vengono mantenuti.

Oggetti `VolumeGroupSnapshot` Kubernetes

Un oggetto Kubernetes `VolumeGroupSnapshot` è una richiesta di creazione di un'istantanea di più volumi. Così come un PVC rappresenta una richiesta fatta da un utente per un volume, un'istantanea di gruppo di volumi è una richiesta fatta da un utente per creare un'istantanea di un PVC esistente.

Quando arriva una richiesta di snapshot di un gruppo di volumi, Trident gestisce automaticamente la creazione dello snapshot di gruppo per i volumi sul backend ed espone lo snapshot creando un oggetto `VolumeGroupSnapshotContent` unico. Puoi creare snapshot da PVC esistenti e utilizzare gli snapshot come DataSource quando crei nuovi PVC.

> Il ciclo di vita di un VolumeGroupSnapshot è indipendente dal PVC di origine: una snapshot persiste anche dopo l'eliminazione del PVC di origine. Quando si elimina un PVC che ha snapshot associate, Trident contrassegna il volume di supporto per questo PVC in stato **Deleting**, ma non lo rimuove completamente. La snapshot del gruppo di volumi viene rimossa quando tutte le snapshot associate vengono eliminate.

Oggetti `VolumeGroupSnapshotContent` Kubernetes

Un oggetto Kubernetes `VolumeGroupSnapshotContent` rappresenta una snapshot di gruppo presa da un volume già fornito. È analogo a `PersistentVolume` e indica una snapshot fornita sul cluster di storage. Analogamente a `PersistentVolumeClaim` e `PersistentVolume` oggetti, quando viene creata una snapshot, l'oggetto `VolumeSnapshotContent` mantiene una mappatura uno-a-uno con l'oggetto `VolumeSnapshot` che ha richiesto la creazione della snapshot.

L' `VolumeGroupSnapshotContent`oggetto contiene dettagli che identificano il gruppo di snapshot, come il `volumeGroupSnapshotHandle` e i singoli volumeSnapshotHandles esistenti sul sistema storage.

Quando arriva una richiesta di snapshot, Trident crea lo snapshot del gruppo di volumi sul backend. Dopo la creazione dello snapshot del gruppo di volumi, Trident configura un `VolumeGroupSnapshotContent` oggetto e quindi espone lo snapshot all'API di Kubernetes.

Oggetti `CustomResourceDefinition` Kubernetes

Le risorse personalizzate di Kubernetes sono endpoint dell'API di Kubernetes definiti dall'amministratore e utilizzati per raggruppare oggetti simili. Kubernetes supporta la creazione di risorse personalizzate per la memorizzazione di una raccolta di oggetti. Puoi ottenere queste definizioni di risorse eseguendo `kubectl get crds`.

Le Custom Resource Definitions (CRDs) e i relativi metadati degli oggetti sono memorizzati da Kubernetes nel suo archivio dei metadati. Questo elimina la necessità di uno store separato per Trident.

Trident utilizza `CustomResourceDefinition` oggetti per preservare l'identità degli oggetti Trident, come i backend Trident, le classi di storage Trident e i volumi Trident. Questi oggetti sono gestiti da Trident. Inoltre, il framework CSI per le snapshot di volume introduce alcune CRD necessarie per definire le snapshot di volume.

I CRD sono un costrutto di Kubernetes. Gli oggetti delle risorse definite sopra sono creati da Trident. Come semplice esempio, quando un backend viene creato usando `tridentctl`, viene creato un oggetto CRD corrispondente `tridentbackends` per il consumo da parte di Kubernetes.

Ecco alcuni punti da tenere a mente sui CRD di Trident:

- Quando Trident viene installato, viene creato un insieme di CRD che possono essere utilizzati come qualsiasi altro tipo di risorsa.
- Quando si disinstalla Trident usando il `tridentctl uninstall` comando, i pod Trident vengono eliminati ma i CRD creati non vengono ripuliti. Fare riferimento a "Disinstalla Trident" per capire come Trident può essere completamente rimosso e riconfigurato da zero.

Trident `StorageClass` oggetti

Trident crea classi di storage corrispondenti per oggetti Kubernetes `StorageClass` che specificano `csi.trident.netapp.io` nel loro campo provisioner. Il nome della classe di storage corrisponde a quello dell'oggetto Kubernetes `StorageClass` che rappresenta.

> Con Kubernetes, questi oggetti vengono creati automaticamente quando un Kubernetes `StorageClass` che utilizza Trident come provisioner viene registrato.

Le classi di archiviazione comprendono una serie di requisiti per i volumi. Trident confronta questi requisiti con

gli attributi presenti in ogni pool di storage; se corrispondono, quel pool di storage è un target valido per il provisioning dei volumi che utilizzano quella classe di archiviazione.

È possibile creare configurazioni di classi di storage per definire direttamente le classi di storage utilizzando l'API REST. Tuttavia, per le distribuzioni Kubernetes, ci aspettiamo che vengano create quando si registrano nuovi oggetti Kubernetes `StorageClass`.

Oggetti backend Trident

I backend rappresentano i provider di storage su cui Trident effettua il provisioning dei volumi; una singola istanza di Trident può gestire qualsiasi numero di backend.

> Questo è uno dei due tipi di oggetti che si creano e si gestiscono da soli. L'altro è l'oggetto Kubernetes `StorageClass`.

Per ulteriori informazioni su come costruire questi oggetti, consultare "configurazione dei backend".

Trident `StoragePool` oggetti

I pool di storage rappresentano le posizioni distinte disponibili per il provisioning su ciascun backend. Per ONTAP, questi corrispondono ad aggregati nelle SVM. Per NetApp HCI/SolidFire, questi corrispondono a bande QoS specificate dall'amministratore. Ogni pool di storage ha una serie di attributi di storage distinti, che definiscono le sue caratteristiche di prestazioni e di protezione dei dati.

A differenza degli altri oggetti qui, i candidati del pool di storage sono sempre scoperti e gestiti automaticamente.

Trident `Volume` oggetti

I volumi sono l'unità di base del provisioning, comprendendo endpoint di backend, come condivisioni NFS e LUN iSCSI e FC. In Kubernetes, questi corrispondono direttamente a `PersistentVolumes`. Quando si crea un volume, assicurarsi che abbia una storage class, che determina dove quel volume può essere fornito, insieme a una dimensione.

> - In Kubernetes, questi oggetti sono gestiti automaticamente. Puoi visualizzarli per vedere cosa ha effettuato il provisioning Trident.
> - Quando si elimina un PV con le relative snapshot, il volume Trident corrispondente viene aggiornato allo stato **Deleting**. Perché il volume Trident venga eliminato, è necessario rimuovere le snapshot del volume.

Una configurazione di volume definisce le proprietà che un volume provisionato deve avere.

Attributo	Tipo	Richiesto	Descrizione
versione	stringa	no	Versione dell'API Trident ("1")
nome	stringa	sì	Nome del volume da creare
storageClass	stringa	sì	Classe di storage da utilizzare per il provisioning del volume

Attributo	Tipo	Richiesto	Descrizione
dimensione	stringa	sì	Dimensione del volume da fornire in byte
protocollo	stringa	no	Tipo di protocollo da utilizzare; "file" o "blocco"
internalName	stringa	no	Nome dell'oggetto sul sistema storage; generato da Trident
cloneSourceVolume	stringa	no	ontap (nas, san) & solidfire-*: Nome del volume da cui clonare
splitOnClone	stringa	no	ontap (nas, san): Separa il clone dal suo genitore
snapshotPolicy	stringa	no	ontap-*: policy di Snapshot da utilizzare
snapshotReserve	stringa	no	ontap-*: Percentuale del volume riservata alle snapshot
exportPolicy	stringa	no	ontap-nas*: Policy di esportazione da utilizzare
snapshotDirectory	bool	no	ontap-nas*: Se la directory snapshot è visibile
unixPermissions	stringa	no	ontap-nas*: Permessi UNIX iniziali
blockSize	stringa	no	solidfire-*: Dimensione del blocco/settore
fileSystem	stringa	no	Tipo di file system
skipRecoveryQueue	stringa	no	Durante l'eliminazione del volume, ignora la coda di ripristino nello storage ed elimina immediatamente il volume.

Trident genera `internalName` quando crea il volume. Questo consiste in due passaggi. Innanzitutto, antepone il prefisso di storage (il prefisso predefinito `trident` o il prefisso nella configurazione del backend) al nome del volume, ottenendo un nome della forma `<prefix>-<volume-name>`. Quindi procede a sanificare il nome, sostituendo i caratteri non consentiti nel backend. Per i backend ONTAP, sostituisce i trattini con i trattini bassi (quindi il nome interno diventa `<prefix>_<volume-name>`). Per i backend Element, sostituisce i trattini bassi con i trattini.

È possibile utilizzare le configurazioni di volume per effettuare il provisioning diretto dei volumi utilizzando l'API REST, ma nelle implementazioni Kubernetes ci aspettiamo che la maggior parte degli utenti utilizzi il metodo standard Kubernetes `PersistentVolumeClaim`. Trident crea questo oggetto volume automaticamente come parte del processo di provisioning.

Trident `Snapshot` oggetti

Le Snapshot sono una copia point-in-time dei volumi, che può essere utilizzata per il provisioning di nuovi volumi o per il ripristino dello stato. In Kubernetes, queste corrispondono direttamente a `VolumeSnapshotContent` oggetti. Ogni Snapshot è associata a un volume, che è la fonte dei dati per la Snapshot.

Ogni `Snapshot` oggetto include le proprietà elencate di seguito:

Attributo	Tipo	Richiesto	Descrizione
versione	Stringa	Sì	Versione dell'API Trident ("1")
nome	Stringa	Sì	Nome dell'oggetto snapshot Trident
internalName	Stringa	Sì	Nome dell'oggetto snapshot Trident sul sistema storage
volumeName	Stringa	Sì	Nome del volume persistente per cui viene creata la snapshot
volumeInternalName	Stringa	Sì	Nome dell'oggetto volume Trident associato sul sistema storage

In Kubernetes, questi oggetti sono gestiti automaticamente. Puoi visualizzarli per vedere cosa ha effettuato il provisioning Trident.

Quando viene creata una richiesta di oggetto Kubernetes `VolumeSnapshot`, Trident funziona creando un oggetto snapshot sul sistema storage di supporto. Il `internalName` di questo oggetto snapshot viene generato combinando il prefisso `snapshot-` con il `UID` dell'oggetto `VolumeSnapshot` (ad esempio, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` e `volumeInternalName` vengono popolati ottenendo i dettagli del volume di supporto.

Oggetto Trident `ResourceQuota`

Il deamonset Trident consuma una `system-node-critical` Priority Class—la Priority Class più alta disponibile in Kubernetes—per garantire che Trident possa identificare e ripulire i volumi durante lo spegnimento controllato dei nodi e consentire ai pod del deamonset Trident di prevaricare i carichi di lavoro con una priorità inferiore nei cluster in cui vi è un'elevata pressione sulle risorse.

A tal fine, Trident impiega un `ResourceQuota` oggetto per garantire che la Priority Class "system-node-critical" sul daemonset Trident sia soddisfatta. Prima della distribuzione e della creazione del daemonset, Trident cerca il `ResourceQuota` oggetto e, se non viene trovato, lo applica.

Se hai bisogno di un maggiore controllo sulla Resource Quota e sulla Priority Class predefinite, puoi generare un `custom.yaml` o configurare l'oggetto `ResourceQuota` utilizzando l'Helm chart.

Quello che segue è un esempio di un oggetto `ResourceQuota` che dà priorità al daemonset Trident.

```
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
```

Per ulteriori informazioni sulle Resource Quotas, consultare "Kubernetes: Quote di risorse".

Pulire `ResourceQuota` se l'installazione non riesce

Nel raro caso in cui l'installazione fallisca dopo che l'oggetto `ResourceQuota` è stato creato, prova prima "disinstallazione" e poi reinstalla.

Se non funziona, rimuovere manualmente l' `ResourceQuota` oggetto.

Rimuovi `ResourceQuota`

Se si preferisce controllare l'allocazione delle risorse, è possibile rimuovere l'oggetto Trident `ResourceQuota` usando il comando:

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

Pod Security Standards (PSS) e Security Context Constraints (SCC)

Kubernetes Pod Security Standards (PSS) e Pod Security Policies (PSP) definiscono i livelli di autorizzazione e limitano il comportamento dei pod. OpenShift Security Context Constraints (SCC) definiscono analogamente le restrizioni dei pod specifiche del OpenShift Kubernetes Engine. Per fornire questa personalizzazione, Trident abilita alcuni permessi durante l'installazione. Le sezioni seguenti illustrano in dettaglio i permessi impostati da Trident.

PSS sostituisce Pod Security Policies (PSP). PSP è stato deprecato in Kubernetes v1.21 e sarà rimosso in v1.25. Per ulteriori informazioni, consultare "Kubernetes: Sicurezza".

Contesto di sicurezza Kubernetes richiesto e campi correlati

Permesso	Descrizione
Privilegiato	CSI richiede che i punti di mount siano bidirezionali, il che significa che il pod del nodo Trident deve eseguire un container privilegiato. Per ulteriori informazioni, fai riferimento a "Kubernetes: Propagazione del mount".
Networking host	Richiesto per il demone iSCSI. `iscsiadm` gestisce i montaggi iSCSI e utilizza la rete host per comunicare con il demone iSCSI.
IPC dell'host	NFS utilizza la comunicazione interprocesso (IPC) per comunicare con il NFSD.
PID dell'host	Richiesto per avviare `rpc-statd` per NFS. Trident interroga i processi host per determinare se `rpc-statd` è in esecuzione prima di montare volumi NFS.

Capacità	La `SYS_ADMIN` capacità è fornita come parte delle capacità predefinite per i container privilegiati. Ad esempio, Docker imposta queste capacità per i container privilegiati:
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

Seccomp	Il profilo Seccomp è sempre "Unconfined" nei container privilegiati; pertanto, non può essere abilitato in Trident.
SELinux	Su OpenShift, i container privilegiati vengono eseguiti nel dominio `spc_t` ("Super Privileged Container") e i container non privilegiati vengono eseguiti nel dominio `container_t`. Su `containerd`, con `container-selinux` installato, tutti i container vengono eseguiti nel dominio `spc_t`, il che disabilita effettivamente SELinux. Pertanto, Trident non aggiunge `seLinuxOptions` ai container.
DAC	I container privilegiati devono essere eseguiti come root. I container non privilegiati vengono eseguiti come root per accedere ai socket unix richiesti da CSI.

Standard di sicurezza dei Pod (PSS)

Etichetta	Descrizione	Predefinito
`pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version`	Consente al Trident Controller e ai nodi di essere ammessi nello spazio dei nomi di installazione. Non modificare l'etichetta dello spazio dei nomi.	`enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>`

La modifica delle etichette degli spazi dei nomi può causare la mancata programmazione dei pod, un "Error creating: …" o "Warning: trident-csi-…". Se ciò accade, verificare se l'etichetta dello spazio dei nomi per `privileged` è stata modificata. In tal caso, reinstallare Trident.

Politiche di sicurezza dei Pod (PSP)

Campo	Descrizione	Predefinito
`allowPrivilegeEscalation`	I container privilegiati devono consentire l'escalation dei privilegi.	`true`
`allowedCSIDrivers`	Trident non utilizza volumi CSI effimeri inline.	Vuoto
`allowedCapabilities`	I container non privilegiati di Trident non richiedono più capacità rispetto al set predefinito e i container privilegiati ricevono tutte le capacità possibili.	Vuoto
`allowedFlexVolumes`	Trident non fa uso di un "Driver FlexVolume", pertanto non sono inclusi nell'elenco dei volumi consentiti.	Vuoto
`allowedHostPaths`	Il pod del nodo Trident monta il filesystem root del nodo, quindi non vi è alcun vantaggio nell'impostare questo elenco.	Vuoto
`allowedProcMountTypes`	Trident non utilizza alcun `ProcMountTypes`.	Vuoto
`allowedUnsafeSysctls`	Trident non richiede alcuna operazione non sicura `sysctls`.	Vuoto
`defaultAddCapabilities`	Non è necessario aggiungere funzionalità ai container privilegiati.	Vuoto
`defaultAllowPrivilegeEscalation`	L'autorizzazione all'escalation dei privilegi viene gestita in ogni pod Trident.	`false`
`forbiddenSysctls`	Non sono consentiti `sysctls`.	Vuoto
`fsGroup`	I container Trident vengono eseguiti come root.	`RunAsAny`
`hostIPC`	Il montaggio dei volumi NFS richiede che l'host IPC comunichi con `nfsd`	`true`
`hostNetwork`	iscsiadm richiede che la rete host comunichi con il demone iSCSI.	`true`
`hostPID`	È necessario il PID host per verificare se `rpc-statd` è in esecuzione sul nodo.	`true`

Campo	Descrizione	Predefinito
`hostPorts`	Trident non utilizza alcuna porta host.	Vuoto
`privileged`	I pod del nodo Trident devono eseguire un container privilegiato per poter montare i volumi.	`true`
`readOnlyRootFilesystem`	I pod del nodo Trident devono scrivere sul file system del nodo.	`false`
`requiredDropCapabilities`	I pod del nodo Trident eseguono un container privilegiato e non possono rilasciare capacità.	`none`
`runAsGroup`	I container Trident vengono eseguiti come root.	`RunAsAny`
`runAsUser`	I container Trident vengono eseguiti come root.	`runAsAny`
`runtimeClass`	Trident non utilizza `RuntimeClasses`.	Vuoto
`seLinux`	Trident non imposta `seLinuxOptions` perché attualmente ci sono differenze nel modo in cui i runtime dei container e le distribuzioni di Kubernetes gestiscono SELinux.	Vuoto
`supplementalGroups`	I container Trident vengono eseguiti come root.	`RunAsAny`
`volumes`	I pod Trident richiedono questi plugin di volume.	`hostPath, projected, emptyDir`

Vincoli del contesto di sicurezza (SCC)

Etichette	Descrizione	Predefinito
`allowHostDirVolumePlugin`	I pod del nodo Trident montano il file system radice del nodo.	`true`
`allowHostIPC`	Il montaggio di volumi NFS richiede che l'host IPC comunichi con `nfsd`.	`true`
`allowHostNetwork`	iscsiadm richiede che la rete host comunichi con il demone iSCSI.	`true`
`allowHostPID`	È necessario il PID host per verificare se `rpc-statd` è in esecuzione sul nodo.	`true`
`allowHostPorts`	Trident non utilizza alcuna porta host.	`false`

Etichette	Descrizione	Predefinito
`allowPrivilegeEscalation`	I container privilegiati devono consentire l'escalation dei privilegi.	`true`
`allowPrivilegedContainer`	I pod del nodo Trident devono eseguire un container privilegiato per poter montare i volumi.	`true`
`allowedUnsafeSysctls`	Trident non richiede alcuna operazione non sicura `sysctls`.	`none`
`allowedCapabilities`	I container non privilegiati di Trident non richiedono più capacità rispetto al set predefinito e i container privilegiati ricevono tutte le capacità possibili.	Vuoto
`defaultAddCapabilities`	Non è necessario aggiungere funzionalità ai container privilegiati.	Vuoto
`fsGroup`	I container Trident vengono eseguiti come root.	`RunAsAny`
`groups`	Questo SCC è specifico per Trident ed è associato al suo utente.	Vuoto
`readOnlyRootFilesystem`	I pod del nodo Trident devono scrivere sul file system del nodo.	`false`
`requiredDropCapabilities`	I pod del nodo Trident eseguono un container privilegiato e non possono rilasciare capacità.	`none`
`runAsUser`	I container Trident vengono eseguiti come root.	`RunAsAny`
`seLinuxContext`	Trident non imposta `seLinuxOptions` perché attualmente ci sono differenze nel modo in cui i runtime dei container e le distribuzioni di Kubernetes gestiscono SELinux.	Vuoto
`seccompProfiles`	I container privilegiati vengono sempre eseguiti come "Unconfined".	Vuoto
`supplementalGroups`	I container Trident vengono eseguiti come root.	`RunAsAny`
`users`	Viene fornita una voce per associare questo SCC all'utente Trident nel namespace Trident.	n/d
`volumes`	I pod Trident richiedono questi plugin di volume.	`hostPath, downwardAPI, projected, emptyDir`