Riferimento

Astra Trident

NetApp
December 03, 2024

Sommario

Riferimento

Porte Astra Trident

Scopri di più sulle porte utilizzate da Astra Trident per le comunicazioni.

Porte Astra Trident

Astra Trident comunica tramite le seguenti porte:

Porta	Scopo
8443	HTTPS backchannel
8001	Endpoint delle metriche Prometheus
8000	Server REST Trident
17546	Porta della sonda liveness/readiness utilizzata dai pod demonset di Trident

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

API REST di Astra Trident

Anche se "comandi e opzioni tridentctl" sono il modo più semplice per interagire con l'API REST di Astra Trident, puoi utilizzare direttamente l'endpoint REST, se preferisci.

Quando utilizzare l'API REST

REST API è utile per le installazioni avanzate che utilizzano Astra Trident come binario standalone nelle implementazioni non Kubernetes.

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

Utilizzo dell'API REST

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

L'API funziona come segue:

OTTIENI

`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 la creazione dell'oggetto. Per le specifiche di ciascun tipo di oggetto, fare riferimento alla "Gestisci Astra 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 riescono a eseguire l'operazione.

ELIMINARE

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

Elimina la risorsa denominata.

> I volumi associati ai backend o alle classi di storage continueranno ad esistere; questi devono essere cancellati separatamente. Per ulteriori informazioni, fare riferimento a "Gestisci Astra Trident usando tridentctl".

Opzioni della riga di comando

Astra Trident espone diverse opzioni della riga di comando per Trident orchestrator. È possibile utilizzare queste opzioni per modificare la distribuzione.

Registrazione

`-debug`

Attiva l'output di debug.

`-loglevel <level>`

Imposta il livello di registrazione (debug, info, warning, error, Fatal). Il valore predefinito è INFO.

Kubernetes

`-k8s_pod`

Utilizza questa opzione o `-k8s_api_server` per attivare il supporto Kubernetes. Questa impostazione fa in modo che Trident utilizzi le credenziali dell'account del servizio Kubernetes del pod che lo contiene per contattare il server API. Questo funziona solo quando Trident viene eseguito come pod in un cluster Kubernetes con account di servizio abilitati.

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

Utilizza questa opzione o `-k8s_pod` per attivare il supporto Kubernetes. Quando specificato, Trident si connette al server API Kubernetes utilizzando l'indirizzo e la porta non sicuri forniti. Ciò consente a Trident di essere implementato all'esterno di un pod; tuttavia, supporta solo connessioni non sicure al server API. Per connettersi in modo sicuro, implementa 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>`

Ascoltare su questa porta piuttosto che un socket di dominio UNIX.

`-config <file>`

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

RIPOSO

`-address <ip-or-host>`

Specifica l'indirizzo in cui il server di GESTIONE DI Trident deve ascoltare. L'impostazione predefinita è localhost. Quando si ascolta 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 accessibile l'interfaccia REST dall'indirizzo IP del pod.

> L'interfaccia REST di Trident può essere configurata per l'ascolto e la distribuzione solo su 127.0.0.1 (per IPv4) o [::1] (per IPv6).

`-port <port-number>`

Specifica la porta sulla quale il server di GESTIONE DI Trident deve ascoltare. Il valore predefinito è 8000.

`-rest`

Attiva l'interfaccia REST. L'impostazione predefinita è true.

Kubernetes e Trident Objects

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

In che modo gli oggetti interagiscono tra loro?

Forse il modo più semplice per comprendere gli oggetti, il loro scopo e il modo in cui interagiscono è seguire una singola richiesta di storage da parte di un utente Kubernetes:

1. Un utente crea un modulo che richiede un `PersistentVolumeClaim` nuovo `PersistentVolume` sistema Kubernetes di una particolare dimensione `StorageClass`, precedentemente configurato dall'amministratore.
2. Kubernetes `StorageClass` identifica Trident come provisioner e include parametri che indicano a Trident come eseguire il provisioning di un volume per la classe richiesta.
3. Trident cerca il proprio `StorageClass` nome con lo stesso nome che identifica la corrispondenza `Backends` e `StoragePools` che può utilizzare per eseguire il provisioning dei volumi per la classe.

4. Trident esegue il provisioning dello storage in 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 lo `PersistentVolume` storage e quello effettivo.
5. Kubernetes associa il `PersistentVolumeClaim` al nuovo `PersistentVolume`. Pod che includono il `PersistentVolumeClaim` montaggio di PersistentVolume su qualsiasi host su cui viene eseguito.
6. Un utente crea uno `VolumeSnapshot` di un PVC esistente, utilizzando un `VolumeSnapshotClass` che punta a Trident.
7. Trident identifica il volume associato al PVC e crea un'istantanea del volume sul backend. Crea inoltre un `VolumeSnapshotContent` che istruisca Kubernetes su come identificare lo snapshot.
8. Un utente può creare un `PersistentVolumeClaim` usando `VolumeSnapshot` come origine.
9. Trident identifica lo snapshot richiesto ed esegue lo stesso insieme di passaggi necessari per la creazione di un `PersistentVolume` e di un `Volume`.

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

Oggetti Kubernetes `PersistentVolumeClaim`

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

Oltre alla specifica standard, Trident consente agli utenti di specificare le seguenti annotazioni specifiche del volume se desiderano sovrascrivere i valori predefiniti impostati nella configurazione di 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, gcp-cvs, 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, gcp-cvs
trident.netapp.io/snapshotDirectory	SnapshotDirectory	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	Dimensione blocco	solidfire-san

Se il PV creato dispone della `Delete` policy di recupero, Trident elimina sia il PV che il volume di backup quando il PV viene rilasciato (ovvero quando l'utente elimina il PVC). In caso di errore dell'azione di eliminazione, Trident contrassegna il PV come tale e riprova periodicamente l'operazione fino a quando non viene eseguita correttamente o finché il PV non viene cancellato manualmente. Se il PV utilizza `Retain` la policy, Trident la ignora e presuppone che l'amministratore lo ripulisca da Kubernetes e dal back-end, consentendo di eseguire il backup o l'ispezione del volume prima della sua rimozione. L'eliminazione del PV non comporta l'eliminazione del volume di backup da parte di Trident. È necessario rimuoverlo utilizzando l'API REST (`tridentctl`).

Trident supporta la creazione di snapshot dei volumi utilizzando la specifica CSI: È possibile creare un'istantanea del volume e utilizzarla come origine dati per clonare i PVC esistenti. In questo modo, le copie point-in-time di PVS possono essere esposte a Kubernetes sotto forma di snapshot. Le istantanee possono quindi essere utilizzate per creare un nuovo PVS. Date un'occhiata a `On-Demand Volume Snapshots` per vedere come funzionerebbe.

Trident fornisce anche le `cloneFromPVC` annotazioni e `splitOnClone` per la creazione dei cloni. È possibile 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 questo set di annotazioni, Trident clona il volume corrispondente al PVC mysql, invece di eseguire il provisioning di un volume da zero.

Considerare i seguenti punti:

- Si consiglia di clonare un volume inattivo.
- Un PVC e il relativo clone devono trovarsi nello stesso spazio dei nomi Kubernetes e avere la stessa classe di storage.
- Con i `ontap-nas` driver e `ontap-san`, potrebbe essere opportuno impostare l'annotazione PVC `trident.netapp.io/splitOnClone` insieme a `trident.netapp.io/cloneFromPVC`. Se `trident.netapp.io/splitOnClone` impostato su `true`, Trident divide il volume clonato dal volume principale e, quindi, disaccoppiando completamente il ciclo di vita del volume clonato dal volume principale a spese di una perdita di efficienza dello storage. La mancata impostazione `trident.netapp.io/splitOnClone` o impostazione in `false` modo da ridurre il consumo di spazio sul backend, a scapito della creazione di dipendenze tra i volumi principale e clone, in modo che il volume principale non possa essere eliminato a meno che il clone non venga eliminato per primo. Uno scenario in cui la suddivisione del clone ha senso è la clonazione di un volume di database vuoto in cui si prevede che il volume e il relativo clone divergano notevolmente e non traggano beneficio dall'efficienza dello storage offerta da ONTAP.

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

Oggetti Kubernetes `PersistentVolume`

Un oggetto Kubernetes `PersistentVolume` rappresenta una parte dello storage resa disponibile per il cluster Kubernetes. Ha un ciclo di vita indipendente dal pod che lo utilizza.

> Trident crea `PersistentVolume` oggetti e li registra automaticamente nel cluster Kubernetes in base ai volumi forniti. Non ci si aspetta di gestirli da soli.

Quando si crea un PVC che fa riferimento a un PVC basato su Trident `StorageClass` , Trident esegue il

provisioning di un nuovo volume utilizzando la classe di archiviazione corrispondente e registra un nuovo PV per quel volume. Nella configurazione del volume sottoposto a provisioning e del PV corrispondente, Trident segue le seguenti regole:

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

Oggetti Kubernetes `StorageClass`

Gli oggetti Kubernetes `StorageClass` sono specificati per nome in `PersistentVolumeClaims` per effettuare il provisioning dello storage con un set di proprietà. La stessa classe di storage identifica il provider da utilizzare e definisce il set di proprietà in termini che il provider riconosce.

Si tratta di 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 è simile al seguente:

```
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 eseguire il provisioning dei volumi per la classe.

I parametri della classe di storage sono:

Attributo	Tipo	Obbligatorio	Descrizione
attributi	map[string]string	no	Vedere la sezione attributi riportata di seguito
StoragePools	map[string]StringList	no	Mappatura dei nomi backend agli elenchi di pool di storage all'interno di
AddtionalStoragePools	map[string]StringList	no	Mappatura dei nomi backend agli elenchi di pool di storage all'interno di

Attributo	Tipo	Obbligatorio	Descrizione
EsclusiveStoragePools	map[string]StringList	no	Mappatura dei nomi backend agli elenchi di pool di storage all'interno di

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

Attributi di selezione del pool di storage

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

Attributo	Tipo	Valori	Offerta	Richiesta	Supportato da
supporti[1]	stringa	hdd, ibrido, ssd	Il pool contiene supporti di questo tipo; ibridi significa entrambi	Tipo di supporto 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	thick: all ONTAP; thin: all ONTAP e solidfire-san
BackendType	stringa	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, gcp-cvs, azure-netapp-files, ontap-san-economy	Il pool appartiene a questo tipo di backend	Backend specificato	Tutti i driver
snapshot	bool	vero, falso	Il pool supporta volumi con snapshot	Volume con snapshot attivate	ontap-nas, ontap-san, solidfire-san, gcp-cvs
cloni	bool	vero, falso	Il pool supporta la clonazione dei volumi	Volume con cloni attivati	ontap-nas, ontap-san, solidfire-san, gcp-cvs
crittografia	bool	vero, falso	Il pool supporta volumi crittografati	Volume con crittografia attivata	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san

Attributo	Tipo	Valori	Offerta	Richiesta	Supportato da
IOPS	int	intero positivo	Il pool è in grado di garantire IOPS in questa gamma	Volume garantito per questi IOPS	solidfire-san

[1]: Non supportato dai sistemi ONTAP Select

Nella maggior parte dei casi, i valori richiesti influiscono direttamente sul provisioning; ad esempio, la richiesta di thick provisioning comporta un volume con provisioning spesso. Tuttavia, un pool di storage di elementi utilizza i valori IOPS minimi e massimi 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, è possibile utilizzare `attributes` da solo per modellare le qualità dello storage necessario per soddisfare le esigenze di una particolare classe. Trident rileva e seleziona automaticamente i pool di storage che corrispondono a *tutti* di `attributes` quelli specificati.

Se non è possibile utilizzare `attributes` per selezionare automaticamente i pool giusti per una classe, è possibile utilizzare i `storagePools` parametri e `additionalStoragePools` per perfezionare ulteriormente i pool o anche per selezionare un set specifico di pool.

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

Puoi utilizzare questo `additionalStoragePools` parametro per estendere il set di pool utilizzati da Trident per il provisioning, indipendentemente dai pool selezionati dai `attributes` parametri e `storagePools`.

È possibile utilizzare questo `excludeStoragePools` parametro per filtrare l'insieme di pool utilizzati da Trident per il provisioning. L'utilizzo di questo parametro consente di rimuovere i pool corrispondenti.

Nei `storagePools` parametri e `additionalStoragePools`, ogni voce assume il formato `<backend>:<storagePoolList>`, dove `<storagePoolList>` è un elenco separato da virgole di pool di archiviazione 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 i valori di backend che per quelli di elenco. Potete usare `tridentctl get backend` per ottenere l'elenco dei backend e dei relativi insiemi.

Attributi Kubernetes

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

Attributo	Tipo	Valori	Descrizione	Driver pertinenti	Versione di 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	Tutto
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, gcp-cvs, azure-netapp-files	1.11+
VolumeBindingMode	stringa	Immediato, WaitForFirstConsumer	Scegliere quando si verifica il binding del volume e il provisioning dinamico	Tutto	1,19 - 1,26

- Il `fsType` parametro viene utilizzato per controllare il tipo di file system desiderato per i LUN SAN. Inoltre, Kubernetes usa anche la presenza di `fsType` in una classe di storage per indicare che esiste un file system. La proprietà del volume può essere controllata utilizzando il `fsGroup` contesto di sicurezza di un pod solo se `fsType` è impostato. Fare riferimento alla "Kubernetes: Consente di configurare un contesto di protezione per un Pod o un container" per una panoramica sull'impostazione della proprietà del volume mediante il `fsGroup` contesto. Kubernetes applicherà il `fsGroup` valore solo se:
  - `fsType` viene impostato nella classe di archiviazione.
  - La modalità di accesso PVC è RWO.

  Per i driver di storage NFS, esiste già un filesystem come parte dell'esportazione NFS. Per poter utilizzare la `fsGroup` classe di archiviazione è comunque necessario specificare un `fsType`. è possibile impostarlo su o su `nfs` qualsiasi valore non nullo.
- Per ulteriori dettagli sull'espansione del volume, fare riferimento alla"Espandere i volumi".
- Il pacchetto del programma di installazione di Trident fornisce diverse definizioni di classi di archiviazione di esempio da utilizzare con Trident in `sample-input/storage-class-*.yaml`. L'eliminazione di una classe di storage Kubernetes comporta l'eliminazione anche della classe di storage Trident corrispondente.

Oggetti Kubernetes `VolumeSnapshotClass`

Gli oggetti Kubernetes `VolumeSnapshotClass` sono analoghi a `StorageClasses`. Consentono di definire più classi di storage e vengono utilizzate dagli snapshot dei volumi per associare lo snapshot alla classe di snapshot richiesta. Ogni snapshot di volume è associato a una singola classe di snapshot di volume.

Un `VolumeSnapshotClass` deve essere definito da un amministratore per creare snapshot. Viene creata una classe di snapshot del volume con la seguente definizione:

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

```
`driver`Specifica in Kubernetes che le richieste di snapshot di volume
della `csi-snapclass` classe sono gestite da Trident.
`deletionPolicy`Specifica l'azione da eseguire quando è necessario
eliminare uno snapshot. Quando `deletionPolicy` è impostato su `Delete`,
gli oggetti snapshot del volume e lo snapshot sottostante nel cluster di
archiviazione vengono rimossi quando viene eliminato uno snapshot. In
alternativa, impostarlo su `Retain` significa che `VolumeSnapshotContent`
e lo snapshot fisico vengono conservati.
```

Oggetti Kubernetes `VolumeSnapshot`

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

Quando arriva una richiesta di snapshot di un volume, Trident gestisce automaticamente la creazione dello snapshot per il volume sul backend ed espone lo snapshot creando un oggetto univoco `VolumeSnapshotContent`. È possibile creare snapshot da PVC esistenti e utilizzarle come DataSource durante la creazione di nuovi PVC.

> Il ciclo di vita di una VolumeSnapshot è indipendente dal PVC di origine: Una snapshot persiste anche dopo la cancellazione del PVC di origine. Quando si elimina un PVC con snapshot associate, Trident contrassegna il volume di backup per questo PVC in uno stato di **eliminazione**, ma non lo rimuove completamente. Il volume viene rimosso quando vengono eliminate tutte le snapshot associate.

Oggetti Kubernetes `VolumeSnapshotContent`

Un oggetto Kubernetes `VolumeSnapshotContent` rappresenta una snapshot ricavata da un volume già sottoposto a provisioning. È analogo a e indica una `PersistentVolume` snapshot sottoposta a provisioning sul cluster di storage. Analogamente agli `PersistentVolumeClaim` oggetti e `PersistentVolume`, quando

viene creato uno snapshot, l' `VolumeSnapshotContent`oggetto mantiene una mappatura uno a uno all' `VolumeSnapshot`oggetto, che aveva richiesto la creazione dello snapshot.

L' `VolumeSnapshotContent`oggetto contiene dettagli che identificano in modo univoco l'istantanea, ad esempio `snapshotHandle` . Si tratta di `snapshotHandle` una combinazione univoca del nome del PV e del nome dell' `VolumeSnapshotContent`oggetto.

Quando arriva una richiesta di snapshot, Trident crea lo snapshot sul back-end. Dopo aver creato la snapshot, Trident configura un `VolumeSnapshotContent` oggetto ed espone quindi la snapshot nell'API Kubernetes.

> In genere, non è necessario gestire l' `VolumeSnapshotContent`oggetto. Un'eccezione a questo è quando si desidera "importare uno snapshot di volume"creare al di fuori di Astra Trident.

Oggetti Kubernetes `CustomResourceDefinition`

Kubernetes Custom Resources sono endpoint dell'API Kubernetes definiti dall'amministratore e utilizzati per raggruppare oggetti simili. Kubernetes supporta la creazione di risorse personalizzate per l'archiviazione di un insieme di oggetti. È possibile ottenere queste definizioni delle risorse eseguendo `kubectl get crds`.

Le definizioni delle risorse personalizzate (CRD) e i relativi metadati degli oggetti associati vengono memorizzati da Kubernetes nel relativo archivio di metadati. Ciò elimina la necessità di un punto vendita separato per Trident.

Astra Trident utilizza `CustomResourceDefinition` gli 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 di snapshot dei volumi CSI introduce alcuni CRD necessari per definire le snapshot dei volumi.

I CRD sono un costrutto Kubernetes. Gli oggetti delle risorse sopra definite vengono creati da Trident. Come semplice esempio, quando un backend viene creato utilizzando `tridentctl`, un oggetto CRD corrispondente `tridentbackends` viene creato per essere utilizzato da Kubernetes.

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

- Una volta installato Trident, viene creato un set di CRD che possono essere utilizzati come qualsiasi altro tipo di risorsa.
- Quando si disinstalla Trident utilizzando il `tridentctl uninstall` comando, i pod Trident vengono eliminati ma i CRD creati non vengono puliti. Fare riferimento a "Disinstallare Trident" per informazioni su come Trident può essere completamente rimosso e riconfigurato da zero.

Oggetti Astra Trident `StorageClass`

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

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

Le classi di storage comprendono un insieme di requisiti per i volumi. Trident abbina questi requisiti agli attributi presenti in ciascun pool di storage; se corrispondono, tale pool di storage è una destinazione valida per il provisioning dei volumi che utilizzano tale classe di storage.

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

Oggetti di backend Astra Trident

I backend rappresentano i provider di storage in cima ai quali Trident esegue il provisioning dei volumi; una singola istanza Trident può gestire qualsiasi numero di backend.

> Si tratta di uno dei due tipi di oggetti creati e gestiti dall'utente. L'altro è l'oggetto Kubernetes `StorageClass`.

Per ulteriori informazioni su come costruire questi oggetti, fare riferimento a "configurazione dei backend".

Oggetti Astra Trident `StoragePool`

I pool di storage rappresentano le diverse posizioni disponibili per il provisioning su ciascun backend. Per ONTAP, questi corrispondono agli aggregati nelle SVM. Per NetApp HCI/SolidFire, queste corrispondono alle bande QoS specificate dall'amministratore. Per Cloud Volumes Service, questi corrispondono alle regioni dei provider di cloud. Ogni pool di storage dispone di un insieme di attributi di storage distinti, che definiscono le caratteristiche di performance e di protezione dei dati.

A differenza degli altri oggetti qui presenti, i candidati del pool di storage vengono sempre rilevati e gestiti automaticamente.

Oggetti Astra Trident `Volume`

I volumi sono l'unità di provisioning di base, che comprende endpoint back-end, come condivisioni NFS e LUN iSCSI. In Kubernetes, questi corrispondono direttamente a `PersistentVolumes`. Quando si crea un volume, assicurarsi che disponga di una classe di storage, che determini la destinazione del provisioning di quel volume, insieme a una dimensione.

> - In Kubernetes, questi oggetti vengono gestiti automaticamente. È possibile visualizzarli per visualizzare il provisioning di Trident.
> - Quando si elimina un PV con snapshot associati, il volume Trident corrispondente viene aggiornato allo stato **Deleting**. Per eliminare il volume Trident, è necessario rimuovere le snapshot del volume.

Una configurazione del volume definisce le proprietà che un volume sottoposto a provisioning deve avere.

Attributo	Tipo	Obbligatorio	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 durante il provisioning del volume

Attributo	Tipo	Obbligatorio	Descrizione
dimensione	stringa	sì	Dimensione del volume per il provisioning in byte
protocollo	stringa	no	Tipo di protocollo da utilizzare; "file" o "blocco"
InternalName (Nome interno)	stringa	no	Nome dell'oggetto sul sistema di storage; generato da Trident
CloneSourceVolume	stringa	no	ONTAP (nas, san) e SolidFire-*: Nome del volume da cui clonare
SplitOnClone	stringa	no	ONTAP (nas, san): Suddividere il clone dal suo padre
SnapshotPolicy	stringa	no	ONTAP-*: Policy di snapshot da utilizzare
SnapshotReserve	stringa	no	ONTAP-*: Percentuale di volume riservato agli snapshot
ExportPolicy	stringa	no	ontap-nas*: Policy di esportazione da utilizzare
SnapshotDirectory	bool	no	ontap-nas*: Indica se la directory di snapshot è visibile
UnixPermissions	stringa	no	ontap-nas*: Autorizzazioni UNIX iniziali
Dimensione blocco	stringa	no	SolidFire-*: Dimensione blocco/settore
Filesystem	stringa	no	Tipo di file system

Trident genera `internalName` durante la creazione del volume. Si tratta di due fasi. Innanzitutto, il prefisso di archiviazione (predefinito o prefisso nella configurazione backend) viene preceduto `trident` dal nome del volume, dando come risultato un nome del form `<prefix>-<volume-name>`. Quindi, procede alla cancellazione del nome, sostituendo i caratteri non consentiti nel backend. Per i backend ONTAP, sostituisce i trattini con i caratteri di sottolineatura (quindi, il nome interno diventa `<prefix>_<volume-name>`). Per i backend degli elementi, sostituisce i caratteri di sottolineatura con trattini.

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

Oggetti Astra Trident `Snapshot`

Gli snapshot sono una copia point-in-time dei volumi, che può essere utilizzata per eseguire il provisioning di nuovi volumi o lo stato di ripristino. In Kubernetes, questi corrispondono direttamente agli

`VolumeSnapshotContent` oggetti. Ogni snapshot è associato a un volume, che è l'origine dei dati per lo snapshot.

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

Attributo	Tipo	Obbligatorio	Descrizione
versione	Stringa	Sì	Versione dell'API Trident ("1")
nome	Stringa	Sì	Nome dell'oggetto snapshot Trident
InternalName (Nome interno)	Stringa	Sì	Nome dell'oggetto snapshot Trident sul sistema di storage
VolumeName	Stringa	Sì	Nome del volume persistente per il quale viene creato lo snapshot
VolumeInternalName	Stringa	Sì	Nome dell'oggetto volume Trident associato nel sistema di storage

In Kubernetes, questi oggetti vengono gestiti automaticamente. È possibile visualizzarli per visualizzare il provisioning di Trident.

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

Oggetto Astra Trident `ResourceQuota`

Il deamonset Trident consuma una `system-node-critical` classe di priorità, la classe di priorità più elevata disponibile in Kubernetes, per garantire che Astra Trident possa identificare e ripulire i volumi in fase di shutdown anomalo del nodo e consentire ai pod di daemonset Trident di prevenire i carichi di lavoro con una priorità più bassa nei cluster in cui c'è una pressione elevata delle risorse.

A tale scopo, Astra Trident utilizza un `ResourceQuota` oggetto per garantire che sia soddisfatta una classe di priorità "system-node-critical" sul daemonset Trident. Prima della distribuzione e della creazione di daemonset, Astra Trident cerca l' `ResourceQuota`oggetto e, se non viene rilevato, lo applica.

Se è necessario un maggiore controllo sulla quota di risorse e sulla classe di priorità predefinite, è possibile generare un `custom.yaml` oggetto o configurarlo `ResourceQuota` utilizzando il grafico Helm.

Di seguito viene riportato un esempio di oggetto `ResourceQuota`che dà priorità al demonset 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 quote di risorse, fare riferimento a "Kubernetes: Quote delle risorse".

Pulire `ResourceQuota` se l'installazione non riesce

Nel raro caso in cui l'installazione non riesca dopo la `ResourceQuota` creazione dell'oggetto, provare prima "disinstallazione in corso"e poi reinstallare.

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

Rimuovere `ResourceQuota`

Se preferisci controllare la tua allocazione di risorse, puoi rimuovere l'oggetto Astra 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 Policy (PSP) definiscono i livelli di autorizzazione e limitano il comportamento dei pod. OpenShift Security Context Constraints (SCC) definisce analogamente la restrizione pod specifica per OpenShift Kubernetes Engine. Per fornire questa personalizzazione, Astra Trident abilita alcune autorizzazioni durante l'installazione. Nelle sezioni seguenti sono descritte in dettaglio le autorizzazioni impostate da Astra Trident.

PSS sostituisce Pod Security Policies (PSP). PSP è stato deprecato in Kubernetes v1.21 e verrà rimosso nella versione 1.25. Per ulteriori informazioni, fare riferimento a "Kubernetes: Sicurezza".

Contesto di sicurezza Kubernetes obbligatorio e campi correlati

Permesso	Descrizione
Privilegiato	CSI richiede che i punti di montaggio siano bidirezionali, il che significa che il pod di nodi Trident deve eseguire un container privilegiato. Per ulteriori informazioni, fare riferimento a "Kubernetes: Propagazione del mount".
Rete host	Necessario per il daemon iSCSI. `iscsiadm` Gestisce i mount iSCSI e utilizza la rete host per comunicare con il demone iSCSI.
IPC host	NFS utilizza la comunicazione interprocesso (IPC) per comunicare con NFSD.
PID host	Necessario per avviare `rpc-statd` NFS. Astra Trident interroga i processi degli host per determinare se `rpc-statd` è in esecuzione prima di montare i volumi NFS.

Funzionalità	La `SYS_ADMIN` funzionalità è fornita come parte delle funzionalità predefinite per i contenitori con privilegi. Ad esempio, Docker imposta queste funzionalità per i container privilegiati:
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

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

Standard di sicurezza Pod (PSS)

Etichetta	Descrizione	Predefinito
`pod-security.kubernetes.io/enforce` `pod-security.kubernetes.io/enforce-version`	Consente di ammettere il controller Trident e i nodi nello spazio dei nomi install. 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 dello spazio dei nomi può causare la mancata pianificazione dei pod, un "errore di creazione: …" Oppure "Warning: trident-csi-…". In questo caso, controllare se l'etichetta dello spazio dei nomi per `privileged` è stata modificata. In tal caso, reinstallare Trident.

Policy di sicurezza Pod (PSP)

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

Campo	Descrizione	Predefinito
`hostPorts`	Trident non utilizza porte host.	Vuoto
`privileged`	I pod di nodi Trident devono eseguire un container privilegiato per poter montare i volumi.	`true`
`readOnlyRootFilesystem`	I pod di nodi Trident devono scrivere nel file system del nodo.	`false`
`requiredDropCapabilities`	I pod di nodi Trident eseguono un container privilegiato e non possono rilasciare funzionalità.	`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 si imposta `seLinuxOptions` perché esistono attualmente differenze nel modo in cui i runtime dei container e le distribuzioni Kubernetes gestiscono SELinux.	Vuoto
`supplementalGroups`	I container Trident vengono eseguiti come root.	`RunAsAny`
`volumes`	I pod Trident richiedono questi plug-in di volume.	`hostPath, projected, emptyDir`

SCC (Security Context Constraints)

Etichette	Descrizione	Predefinito
`allowHostDirVolumePlugin`	I pod di nodi Trident montano il filesystem root del nodo.	`true`
`allowHostIPC`	Il montaggio di volumi NFS richiede che l'IPC host comunichi con `nfsd`.	`true`
`allowHostNetwork`	Iscsiadm richiede che la rete host comunichi con il daemon iSCSI.	`true`
`allowHostPID`	Il PID host è necessario per controllare se `rpc-statd` è in esecuzione sul nodo.	`true`
`allowHostPorts`	Trident non utilizza porte host.	`false`
`allowPrivilegeEscalation`	I container con privilegi devono consentire l'escalation dei privilegi.	`true`

Etichette	Descrizione	Predefinito
`allowPrivilegedContainer`	I pod di nodi Trident devono eseguire un container privilegiato per poter montare i volumi.	`true`
`allowedUnsafeSysctls`	Trident non richiede alcuna non sicura `sysctls`.	`none`
`allowedCapabilities`	I container Trident non con privilegi non richiedono più funzionalità rispetto al set predefinito e ai container con privilegi vengono concesse tutte le funzionalità possibili.	Vuoto
`defaultAddCapabilities`	Non è necessario aggiungere funzionalità ai container con privilegi.	Vuoto
`fsGroup`	I container Trident vengono eseguiti come root.	`RunAsAny`
`groups`	Questo SCC è specifico di Trident ed è vincolato al proprio utente.	Vuoto
`readOnlyRootFilesystem`	I pod di nodi Trident devono scrivere nel file system del nodo.	`false`
`requiredDropCapabilities`	I pod di nodi Trident eseguono un container privilegiato e non possono rilasciare funzionalità.	`none`
`runAsUser`	I container Trident vengono eseguiti come root.	`RunAsAny`
`seLinuxContext`	Trident non si imposta `seLinuxOptions` perché esistono attualmente differenze nel modo in cui i runtime dei container e le distribuzioni Kubernetes gestiscono SELinux.	Vuoto
`seccompProfiles`	I container privilegiati vengono sempre eseguiti "senza confinare".	Vuoto
`supplementalGroups`	I container Trident vengono eseguiti come root.	`RunAsAny`
`users`	Viene fornita una voce per associare SCC all'utente Trident nello spazio dei nomi Trident.	n/a.
`volumes`	I pod Trident richiedono questi plug-in di volume.	`hostPath, downwardAPI, projected, emptyDir`

Informazioni sul copyright

Informazioni sul marchio commerciale