Documentazione di Astra Trident 22.01

Astra Trident

NetApp
April 16, 2024

Sommario

Documentazione di Astra Trident 22.01 . . . 1
Note di rilascio . . . 2
- Novità del 22.01.1 . . . 2
- Cambiamenti nel 22.01.0 (dal 22.10.1) . . . 2
- Modifiche in Astra Trident 21.10.1 . . . 3
- Cambiamenti nel 21.10.0 (da Astra Trident 21.07) . . . 3
- Problemi noti . . . 4
- Trova ulteriori informazioni . . . 5

Concetti . . . 6
- Introduzione ad Astra Trident . . . 6
- Driver ONTAP . . . 7
- Provisioning . . . 7
- Snapshot dei volumi . . . 8
- Pool di storage virtuali . . . 9
- Gruppi di accesso ai volumi . . . 10

Inizia subito . . . 11
- Provalo . . . 11
- Requisiti . . . 11
- Panoramica dell’implementazione . . . 15
- Implementazione con l’operatore Trident . . . 18
- Implementare con tridentctl . . . 27
- Cosa succederà? . . . 30

Gestire Astra Trident . . . 36
- Aggiorna Astra Trident . . . 36
- Eseguire l’upgrade con l’operatore . . . 37
- Upgrade con tridentctl . . . 45
- Disinstallare Astra Trident . . . 48
- Downgrade di Astra Trident . . . 50

Utilizzare Astra Trident . . . 54
- Configurare i backend . . . 54
- Crea backend con kubectl . . . 122
- Eseguire la gestione del back-end con kubectl . . . 129
- Eseguire la gestione back-end con tridentctl . . . 130
- Passare da un’opzione di gestione back-end all’altra . . . 132
- Gestire le classi di storage . . . 138
- Eseguire operazioni sui volumi . . . 140
- Preparare il nodo di lavoro . . . 165
- Preparazione automatica del nodo di lavoro . . . 169
- Monitorare Astra Trident . . . 169

Astra Trident per Docker . . . 174
- Prerequisiti per l’implementazione . . . 174
- Implementare Astra Trident . . . 177
- Aggiornare o disinstallare Astra Trident . . . 181

Lavorare con i volumi . . . 183
Raccogliere i log . . . 192
Gestire più istanze di Astra Trident . . . 193
Opzioni di configurazione dello storage . . . 194
Problemi noti e limitazioni . . . 210
Domande frequenti . . . 212
Domande generali . . . 212
Installare e utilizzare Astra Trident su un cluster Kubernetes . . . 212
Risoluzione dei problemi e supporto . . . 214
Aggiorna Astra Trident . . . 215
Gestione di back-end e volumi . . . 215
Supporto . . . 221
Risoluzione dei problemi . . . 222
Risoluzione dei problemi generali . . . 222
Risoluzione dei problemi di un'implementazione Trident non riuscita utilizzando l'operatore . . . 224
Risoluzione dei problemi di un'implementazione Trident non riuscita utilizzando `tridentctl` . . . 226
Best practice e consigli . . . 227
Implementazione . . . 227
Configurazione dello storage . . . 227
Integrare Astra Trident . . . 234
Protezione dei dati . . . 245
Sicurezza . . . 250
Riferimento . . . 252
Porte Astra Trident . . . 252
API REST di Astra Trident . . . 252
Opzioni della riga di comando . . . 253
Prodotti NetApp integrati con Kubernetes . . . 254
Kubernetes e Trident Objects . . . 255
comandi e opzioni tridentctl . . . 266
Versioni precedenti della documentazione . . . 272
Note legali . . . 273
Copyright . . . 273
Marchi . . . 273
Brevetti . . . 273
Direttiva sulla privacy . . . 273
Open source . . . 273

Documentazione di Astra Trident 22.01

Note di rilascio

Le Note di rilascio forniscono informazioni su nuove funzionalità, miglioramenti e correzioni di bug nell'ultima versione di Astra Trident.

> Il `tridentctl` Il file binario per Linux fornito nel file zip del programma di installazione è la versione testata e supportata. Tenere presente che il `macos` binario fornito in `/extras` parte del file zip non è testata o supportata.

Novità del 22.01.1

NetApp continua a migliorare e migliorare i propri prodotti e servizi. Ecco alcune delle funzionalità più recenti di Astra Trident. Per le release precedenti, vedere "Versioni precedenti della documentazione".

> Se si esegue l'aggiornamento da una release precedente di Trident e si utilizza Azure NetApp Files, il `location` il parametro di configurazione è ora un campo singleton obbligatorio.

Correzioni

- Risolto il problema di annullamento della pubblicazione dei volumi sui nodi cancellati. ("Numero GitHub 691")
- Risolto il problema dell'accesso ai campi nil per lo spazio aggregato nelle risposte API ONTAP.

Cambiamenti nel 22.01.0 (dal 22.10.1)

Correzioni

- **Kubernetes:** aumenta il tempo di tentativi di backoff per la registrazione dei nodi per cluster di grandi dimensioni.
- Risolto il problema per cui il driver Azure-netapp-Files poteva essere confuso da più risorse con lo stesso nome.
- Le LIF dati ONTAP SAN IPv6 ora funzionano se specificate con parentesi quadre.
- Risolto il problema a causa del quale il tentativo di importare un volume già importato restituisce EOF lasciando PVC in stato di attesa. ("Numero GitHub 489")
- Risolto il problema relativo al rallentamento delle prestazioni di Astra Trident quando vengono creati più di 32 snapshot su un volume SolidFire.
- Ha sostituito SHA-1 con SHA-256 nella creazione del certificato SSL.
- Driver ANF fisso per consentire nomi di risorse duplicati e limitare le operazioni a una singola posizione.
- Driver ANF fisso per consentire nomi di risorse duplicati e limitare le operazioni a una singola posizione.

Miglioramenti

- Miglioramenti di Kubernetes:
  - Aggiunto supporto per Kubernetes 1.23.
  - Aggiungi le opzioni di pianificazione per i pod Trident se installati tramite Trident Operator o Helm. ("Numero GitHub 651")

- Consenti volumi cross-area nel driver GCP. ("Numero GitHub 633")
- Aggiunto supporto per l'opzione 'unixPermissions' ai volumi ANF. ("Numero GitHub 666")

Dipendenze

L'interfaccia REST di Trident può ascoltare e servire solo a 127.0.0.1 o [::1] indirizzi

Modifiche in Astra Trident 21.10.1

> La versione v21.10.0 presenta un problema che può mettere il controller Trident in uno stato CrashLoopBackOff quando un nodo viene rimosso e quindi aggiunto di nuovo al cluster Kubernetes. Questo problema è stato risolto in v21.10.1 (problema di GitHub 669).

Correzioni

- Correzione della potenziale condizione di gara durante l'importazione di un volume su un backend CVS GCP, con conseguente mancata importazione.
- Risolto un problema che può portare il controller Trident in uno stato CrashLoopBackOff quando un nodo viene rimosso e quindi aggiunto di nuovo al cluster Kubernetes (problema GitHub 669).
- Risolto il problema a causa del quale le SVM non venivano più rilevate se non è stato specificato alcun nome SVM (problema di GitHub 612).

Cambiamenti nel 21.10.0 (da Astra Trident 21.07)

Correzioni

- Risolto il problema a causa del quale i cloni dei volumi XFS non potevano essere montati sullo stesso nodo del volume di origine (problema di GitHub 514).
- Risolto il problema a causa del quale Astra Trident ha registrato un errore irreversibile durante lo shutdown (problema di GitHub 597).
- Correzioni relative a Kubernetes:
  - Restituisce lo spazio utilizzato di un volume come restoreDim minimo quando si creano snapshot con `ontap-nas` e. `ontap-nas-flexgroup` Driver (problema GitHub 645).
  - Risolto il problema in cui `Failed to expand filesystem` L'errore è stato registrato dopo il ridimensionamento del volume (problema di GitHub 560).
  - Risolto il problema di blocco di un pod `Terminating` (Problema 572 di GitHub).
  - Risolto il caso in cui un `ontap-san-economy` FlexVol potrebbe essere pieno di LUN snapshot (problema GitHub 533).
  - Risolto il problema del programma di installazione YAML personalizzato con immagini diverse (problema GitHub 613).
  - Corretto il calcolo delle dimensioni dello snapshot (problema di GitHub 611).
  - Risolto il problema per cui tutti gli installatori di Astra Trident potevano identificare Kubernetes semplici come OpenShift (problema di GitHub 639).
  - Risolto il problema dell'operatore Trident per interrompere la riconciliazione se il server API Kubernetes non è raggiungibile (problema di GitHub 599).

Miglioramenti

- Supporto aggiunto per `unixPermissions` Opzione per volumi di performance GCP-CVS.
- Supporto aggiunto per volumi CVS ottimizzati per la scalabilità in GCP nell'intervallo da 600 GiB a 1 TIB.
- Miglioramenti relativi a Kubernetes:
  - Aggiunto supporto per Kubernetes 1.22.
  - Ha consentito all'operatore Trident e al grafico Helm di lavorare con Kubernetes 1.22 (problema GitHub 628).
  - Aggiunta immagine operatore a. `tridentctl` Comando Images (problema GitHub 570).

Miglioramenti sperimentali

- Aggiunto supporto per la replica dei volumi in `ontap-san` driver.
- Aggiunto il supporto REST di **TECH preview** per `ontap-nas-flexgroup`, `ontap-san`, e. `ontap-nas-economy` driver.

Problemi noti

I problemi noti identificano i problemi che potrebbero impedire l'utilizzo corretto del prodotto.

- Astra Trident ora impone un vuoto `fsType` (`fsType=""`) per i volumi che non dispongono di `fsType` Specificato nella loro StorageClass. Quando si lavora con Kubernetes 1.17 o versioni successive, Trident supporta la fornitura di un vuoto `fsType` Per volumi NFS. Per i volumi iSCSI, è necessario impostare `fsType` Sulla StorageClass quando si applica un `fsGroup` Utilizzo di un contesto di protezione.
- Quando si utilizza un backend tra più istanze di Astra Trident, ciascun file di configurazione backend deve avere un file diverso `storagePrefix` Valore per backend ONTAP o utilizzare un altro `TenantName` Per backend SolidFire. Astra Trident non è in grado di rilevare i volumi creati da altre istanze di Astra Trident. Il tentativo di creare un volume esistente su backend ONTAP o SolidFire ha esito positivo, perché Astra Trident considera la creazione del volume come un'operazione di idempotent. Se `storagePrefix` oppure `TenantName` non differire, potrebbero esserci collisioni di nomi per i volumi creati sullo stesso backend.
- Durante l'installazione di Astra Trident (utilizzando `tridentctl` O l'operatore Trident) e utilizzando `tridentctl` Per gestire Astra Trident, è necessario assicurarsi di `KUBECONFIG` variabile di ambiente impostata. Questo è necessario per indicare il cluster Kubernetes che `tridentctl` dovrebbe lavorare contro. Quando si lavora con ambienti Kubernetes multipli, è necessario assicurarsi che il `KUBECONFIG` il file viene fornito in modo accurato.
- Per eseguire la rigenerazione dello spazio online per iSCSI PVS, il sistema operativo sottostante sul nodo di lavoro potrebbe richiedere il passaggio delle opzioni di montaggio al volume. Questo vale per le istanze RHEL/RedHat CoreOS, che richiedono `discard` "opzione di montaggio"; Assicurarsi che il modello Discard mountOption sia incluso nel[`StorageClass`^] per supportare lo scarto del blocco online.
- Se si dispone di più istanze di Astra Trident per cluster Kubernetes, Astra Trident non è in grado di comunicare con altre istanze e non è in grado di rilevare altri volumi creati, il che comporta un comportamento imprevisto e non corretto se più di un'istanza viene eseguita all'interno di un cluster. Dovrebbe essere presente una sola istanza di Astra Trident per cluster Kubernetes.
- Se basato su Astra Trident `StorageClass` Gli oggetti vengono cancellati da Kubernetes mentre Astra Trident è offline, Astra Trident non rimuove le classi di storage corrispondenti dal proprio database quando torna online. È necessario eliminare queste classi di storage utilizzando `tridentctl` O l'API REST.

- Se un utente elimina un PV fornito da Astra Trident prima di eliminare il PVC corrispondente, Astra Trident non elimina automaticamente il volume di backup. Rimuovere il volume tramite `tridentctl` O l'API REST.
- ONTAP non è in grado di eseguire contemporaneamente il provisioning di più FlexGroup alla volta, a meno che il set di aggregati non sia univoco per ogni richiesta di provisioning.
- Quando si utilizza Astra Trident su IPv6, è necessario specificare `managementLIF` e. `dataLIF` nella definizione di backend tra parentesi quadre. Ad esempio, `[fd20:8b1e:b258:2000:f816:3eff:feec:0]`.
- Se si utilizza `solidfire-san` Driver con OpenShift 4.5, assicurarsi che i nodi di lavoro sottostanti utilizzino MD5 come algoritmo di autenticazione CHAP.

Trova ulteriori informazioni

- "Astra Trident GitHub"
- "Blog di Astra Trident"

Concetti

Introduzione ad Astra Trident

Astra Trident è un progetto open source completamente supportato gestito da NetApp come parte di "Famiglia di prodotti Astra". È stato progettato per soddisfare le esigenze di persistenza delle applicazioni containerizzate utilizzando interfacce standard di settore, come l'interfaccia di storage container (CSI).

Astra Trident si implementa nei cluster Kubernetes come pod e fornisce servizi di orchestrazione dello storage dinamico per i carichi di lavoro Kubernetes. Consente alle applicazioni containerizzate di consumare in modo rapido e semplice storage persistente dall'ampio portfolio NetApp che include ONTAP (AFF/FAS/selezione/cloud/Amazon FSX per NetApp ONTAP), software Element (NetApp HCI/SolidFire), archivio dati Astra, nonché il servizio Azure NetApp Files e Cloud Volumes Service su Google Cloud.

Astra Trident è anche una tecnologia di base per Astra di NetApp, che si occupa di protezione dei dati, disaster recovery, portabilità e casi di utilizzo della migrazione per i carichi di lavoro Kubernetes sfruttando la tecnologia di gestione dei dati leader del settore di NetApp per snapshot, backup, replica e cloning.

Architetture cluster Kubernetes supportate

Astra Trident è supportato con le seguenti architetture Kubernetes:

Kubernetes architetture di cluster	Supportato	Installazione predefinita
Singolo master, calcolo	Sì	Sì
Master multipli, calcolo	Sì	Sì
Master, `etcd`calcolo	Sì	Sì
Master, infrastruttura, calcolo	Sì	Sì

Che cos'è Astra?

Astra semplifica la gestione, la protezione e lo spostamento dei carichi di lavoro containerizzati ricchi di dati eseguiti su Kubernetes all'interno e tra cloud pubblici e on-premise. Astra fornisce e fornisce storage container persistente utilizzando Astra Trident del comprovato e esteso portfolio di storage NetApp nel cloud pubblico e on-premise. Offre inoltre una serie completa di funzionalità avanzate di gestione dei dati applicative, come snapshot, backup e ripristino, log di attività e cloning attivo per la protezione dei dati, disaster recovery/data, audit dei dati e casi di utilizzo della migrazione per i carichi di lavoro Kubernetes.

Puoi iscriverti per una prova gratuita nella pagina Astra.

Per ulteriori informazioni

- "Famiglia di prodotti NetApp Astra"
- "Documentazione del servizio Astra Control"
- "Documentazione di Astra Control Center"

- "Documentazione di Astra Data Store"
- "Documentazione API Astra"

Driver ONTAP

Astra Trident offre cinque driver di storage ONTAP esclusivi per la comunicazione con i cluster ONTAP. Scopri di più su come ciascun driver gestisce la creazione di volumi, il controllo degli accessi e le relative funzionalità.

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
`ontap-nas`	NFS	Filesystem	RWO, RWX, ROX	"", nfs
`ontap-nas-economy`	NFS	Filesystem	RWO, RWX, ROX	"", nfs
`ontap-nas-flexgroup`	NFS	Filesystem	RWO, RWX, ROX	"", nfs
`ontap-san`	ISCSI	Blocco	RWO, ROX, RWX	Nessun filesystem. Dispositivo a blocchi raw
`ontap-san`	ISCSI	Filesystem	RWO, ROX	`xfs`, `ext3`, `ext4`
`ontap-san-economy`	ISCSI	Blocco	RWO, ROX, RWX	Nessun filesystem. Dispositivo a blocchi raw
`ontap-san-economy`	ISCSI	Filesystem	RWO, ROX	`xfs`, `ext3`, `ext4`

I backend ONTAP possono essere autenticati utilizzando le credenziali di accesso per un ruolo di sicurezza (nome utente/password) o utilizzando la chiave privata e il certificato installati sul cluster ONTAP. È possibile aggiornare i backend esistenti per passare da una modalità di autenticazione all'altra con `tridentctl update backend`.

Provisioning

Il provisioning in Astra Trident prevede due fasi principali. La prima fase associa una classe di storage all'insieme di pool di storage di back-end adatti e si verifica come preparazione necessaria prima del provisioning. La seconda fase include la creazione del volume e richiede la scelta di un pool di storage tra quelli associati alla classe di storage del volume in sospeso.

L'associazione dei pool di storage back-end a una classe di storage si basa sugli attributi richiesti dalla classe di storage e su ITS `storagePools`, `additionalStoragePools`, e. `excludeStoragePools` elenchi. Quando si crea una classe di storage, Trident confronta gli attributi e i pool offerti da ciascun backend con quelli richiesti dalla classe di storage. Se gli attributi e il nome di un pool di storage corrispondono a tutti gli attributi e i nomi dei pool richiesti, Astra Trident aggiunge tale pool di storage all'insieme di pool di storage adatti per tale classe di storage. Inoltre, Astra Trident aggiunge tutti i pool di storage elencati in

`additionalStoragePools` di quel set, anche se i relativi attributi non soddisfano tutti o nessuno degli attributi richiesti dalla classe di storage. Utilizzare il `excludeStoragePools` elenco per eseguire l'override e rimuovere i pool di storage dall'utilizzo per una classe di storage. Astra Trident esegue un processo simile ogni volta che si aggiunge un nuovo backend, verificando se i pool di storage soddisfano quelli delle classi di storage esistenti e rimuovendo quelli contrassegnati come esclusi.

Astra Trident utilizza quindi le associazioni tra classi di storage e pool di storage per determinare dove eseguire il provisioning dei volumi. Quando si crea un volume, Astra Trident ottiene prima l'insieme di pool di storage per la classe di storage di quel volume, Inoltre, se si specifica un protocollo per il volume, Astra Trident rimuove i pool di storage che non possono fornire il protocollo richiesto (ad esempio, un backend NetApp HCI/SolidFire non può fornire un volume basato su file mentre un backend NAS ONTAP non può fornire un volume basato su blocchi). Astra Trident crea una sequenza casuale dell'ordine di questo set risultante, per facilitare una distribuzione uniforme dei volumi e quindi lo itera, tentando di eseguire il provisioning del volume su ciascun pool di storage a turno. Se riesce su uno, ritorna con successo, registrando gli eventuali errori riscontrati nel processo. Astra Trident restituisce un errore **solo se** non riesce a eseguire il provisioning su **tutti** i pool di storage disponibili per la classe di storage e il protocollo richiesti.

Snapshot dei volumi

Scopri di più su come Astra Trident gestisce la creazione di snapshot di volumi per i suoi driver.

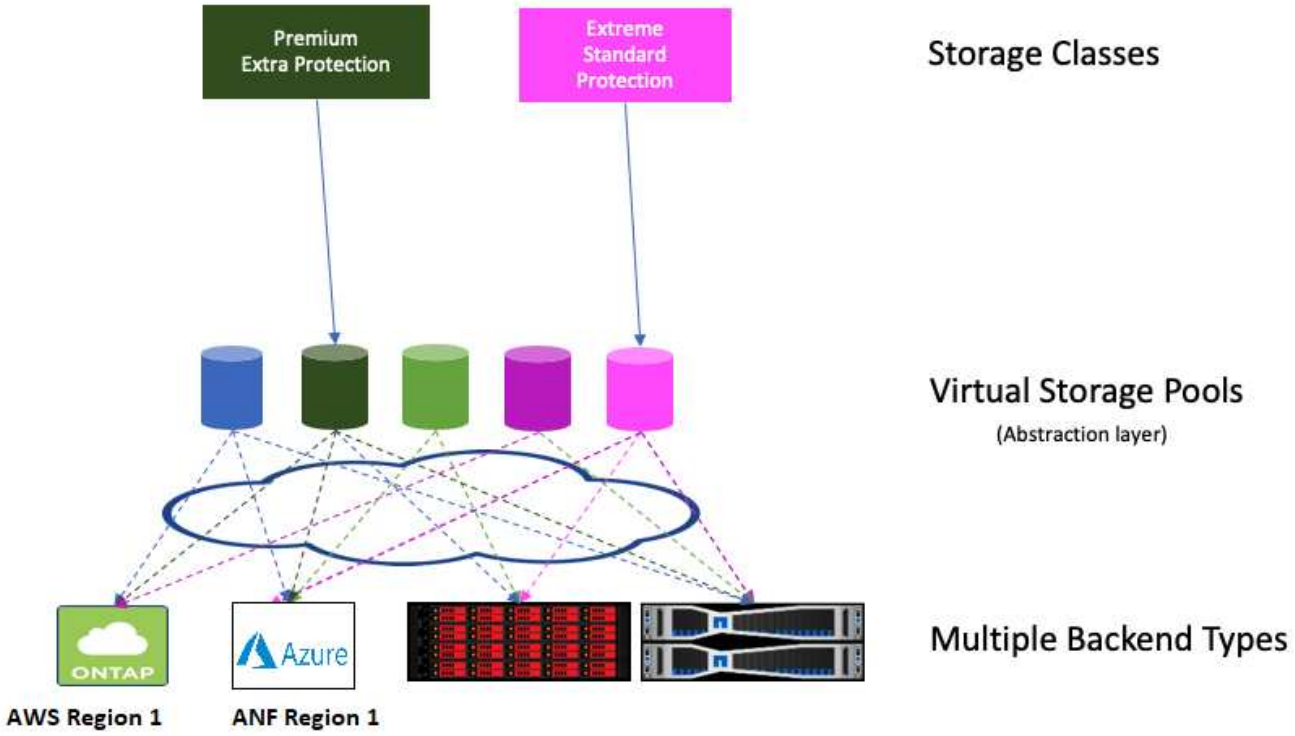
- Per `ontap-nas`, `ontap-san`, `gcp-cvs`, e. `azure-netapp-files` Driver, ogni volume persistente (PV) viene mappato su un FlexVol. Di conseguenza, le snapshot dei volumi vengono create come snapshot NetApp. La tecnologia Snapshot di NetApp offre maggiore stabilità, scalabilità, ripristinabilità e performance rispetto alle tecnologie Snapshot della concorrenza. Queste copie Snapshot sono estremamente efficienti sia nel tempo necessario per crearle che nello spazio di storage.
- Per `ontap-nas-flexgroup` Driver, ogni volume persistente (PV) viene mappato su un FlexGroup. Di conseguenza, le snapshot dei volumi vengono create come snapshot NetApp FlexGroup. La tecnologia Snapshot di NetApp offre maggiore stabilità, scalabilità, ripristinabilità e performance rispetto alle tecnologie Snapshot della concorrenza. Queste copie Snapshot sono estremamente efficienti sia nel tempo necessario per crearle che nello spazio di storage.
- Per `ontap-san-economy` Driver, PVS mappano le LUN create su FlexVol condivisi. VolumeSnapshots di PVS si ottengono eseguendo FlexClone del LUN associato. La tecnologia FlexClone di ONTAP consente di creare copie anche dei set di dati più grandi quasi istantaneamente. Le copie condividono i blocchi di dati con i genitori, senza consumare storage ad eccezione di quanto richiesto per i metadati.
- Per `solidfire-san` Driver, ogni PV viene mappato su un LUN creato nel software NetApp Element/cluster NetApp HCI. Le istantanee Volumesono rappresentate da snapshot degli elementi del LUN sottostante. Queste snapshot sono copie point-in-time e occupano solo una piccola quantità di risorse e spazio di sistema.
- Quando si lavora con `ontap-nas` e. `ontap-san` Driver, le snapshot ONTAP sono copie point-in-time del FlexVol e occupano spazio sul FlexVol stesso. Ciò può comportare una riduzione dello spazio scrivibile nel volume durante la creazione/pianificazione delle istantanee. Un modo semplice per risolvere questo problema consiste nell'aumentare il volume ridimensionandolo tramite Kubernetes. Un'altra opzione consiste nell'eliminare gli snapshot non più necessari. Quando un'istantanea Volumecreata tramite Kubernetes viene eliminata, Astra Trident elimina l'istantanea ONTAP associata. È possibile eliminare anche gli snapshot ONTAP non creati tramite Kubernetes.

Con Astra Trident, puoi utilizzare VolumeSnapshots per creare nuovi PVS da essi. La creazione di PVS da queste snapshot viene eseguita utilizzando la tecnologia FlexClone per i backend ONTAP e CVS supportati. Quando si crea un PV da uno snapshot, il volume di backup è un FlexClone del volume padre dello snapshot. Il `solidfire-san` Il driver utilizza cloni di volumi software Element per creare PVS dalle snapshot. Qui viene creato un clone dallo snapshot degli elementi.

Pool di storage virtuali

I pool di storage virtuali offrono un livello di astrazione tra i backend di storage di Astra Trident e Kubernetes `StorageClasses`. Consentono a un amministratore di definire aspetti quali posizione, performance e protezione per ciascun backend in modo comune e indipendente dal backend senza creare un `StorageClass` specificare il tipo di backend fisico, pool di backend o backend da utilizzare per soddisfare i criteri desiderati.

L'amministratore dello storage può definire pool di storage virtuali su qualsiasi backend Astra Trident in un file di definizione JSON o YAML.



Qualsiasi aspetto specificato al di fuori dell'elenco dei pool virtuali è globale per il backend e verrà applicato a tutti i pool virtuali, mentre ciascun pool virtuale potrebbe specificare uno o più aspetti singolarmente (sovrascrivendo qualsiasi aspetto globale di backend).

> Quando si definiscono i pool di storage virtuali, non tentare di riorganizzare l'ordine dei pool virtuali esistenti in una definizione di back-end. Si consiglia inoltre di non modificare/modificare gli attributi di un pool virtuale esistente e di non definire un nuovo pool virtuale.

La maggior parte degli aspetti è specificata in termini specifici del back-end. Fondamentalmente, i valori di aspetto non sono esposti al di fuori del driver del backend e non sono disponibili per la corrispondenza in `StorageClasses`. L'amministratore definisce invece una o più etichette per ogni pool virtuale. Ogni etichetta è una coppia chiave:valore e le etichette potrebbero essere comuni tra backend univoci. Come per gli aspetti, le etichette possono essere specificate per pool o globali per backend. A differenza degli aspetti, che hanno nomi e valori predefiniti, l'amministratore può definire i valori e le chiavi dell'etichetta in base alle esigenze.

R `StorageClass` identifica il pool virtuale da utilizzare facendo riferimento alle etichette all'interno di un parametro di selezione. I selettori del pool virtuale supportano i seguenti operatori:

Operatore	Esempio	Il valore dell'etichetta di un pool deve:
`=`	performance=premium	Corrispondenza
`!=`	performance!=estrema	Non corrisponde
`in`	posizione in (est, ovest)	Essere nel set di valori
`notin`	performance notin (argento, bronzo)	Non essere nel set di valori
`<key>`	protezione	Esiste con qualsiasi valore
`!<key>`	!protezione	Non esiste

Gruppi di accesso ai volumi

Scopri di più sull'utilizzo di Astra Trident "gruppi di accesso ai volumi".

> Ignorare questa sezione se si utilizza CHAP, che è consigliabile per semplificare la gestione ed evitare il limite di scalabilità descritto di seguito. Inoltre, se si utilizza Astra Trident in modalità CSI, è possibile ignorare questa sezione. Astra Trident utilizza CHAP quando viene installato come provisioning CSI avanzato.

Astra Trident può utilizzare i gruppi di accesso ai volumi per controllare l'accesso ai volumi forniti dall'IT. Se CHAP è disattivato, si aspetta di trovare un gruppo di accesso chiamato `trident` A meno che non si specifichi uno o più ID gruppo di accesso nella configurazione.

Mentre Astra Trident associa nuovi volumi ai gruppi di accesso configurati, non crea o gestisce in altro modo i gruppi di accesso stessi. I gruppi di accesso devono esistere prima che il backend dello storage venga aggiunto ad Astra Trident e devono contenere gli IQN iSCSI di ogni nodo del cluster Kubernetes che potrebbero potenzialmente montare i volumi forniti da tale backend. Nella maggior parte delle installazioni, che include ogni nodo di lavoro nel cluster.

Per i cluster Kubernetes con più di 64 nodi, è necessario utilizzare più gruppi di accesso. Ciascun gruppo di accesso può contenere fino a 64 IQN e ciascun volume può appartenere a quattro gruppi di accesso. Con un massimo di quattro gruppi di accesso configurati, qualsiasi nodo di un cluster di dimensioni fino a 256 nodi potrà accedere a qualsiasi volume. Per gli ultimi limiti sui gruppi di accesso ai volumi, vedere "qui".

Se si sta modificando la configurazione da una che sta utilizzando l'impostazione predefinita `trident` Il gruppo di accesso a uno che utilizza anche altri, include l'ID per `trident` gruppo di accesso nell'elenco.

Inizia subito

Provalo

NetApp fornisce un'immagine di laboratorio pronta all'uso che è possibile richiedere tramite "Test drive di NetApp". Il Test Drive offre un ambiente sandbox dotato di un cluster Kubernetes a tre nodi e Astra Trident installato e configurato. È un ottimo modo per familiarizzare con Astra Trident ed esplorarne le funzionalità.

Un'altra opzione consiste nel vedere "Guida all'installazione di kubeadm" Fornito da Kubernetes.

> Non utilizzare il cluster Kubernetes creato utilizzando queste istruzioni in produzione. Utilizza le guide all'implementazione in produzione fornite dalla tua distribuzione per creare cluster pronti per la produzione.

Se questa è la prima volta che utilizzi Kubernetes, familiarizza con i concetti e gli strumenti "qui".

Requisiti

Inizia esaminando i frontend, i backend e la configurazione host supportati.

> Per ulteriori informazioni sulle porte utilizzate da Astra Trident, vedere "qui".

Frontend supportati (orchestratori)

Astra Trident supporta diversi motori e orchestratori di container, tra cui:

- Anthos on-Prem (VMware) e anthos on Bare Metal 1.8, 1.9, 1.10
- Kubernetes 1.17 o versione successiva (ultimo: 1.23)
- Motore di Mirantis Kubernetes 3.4
- OpenShift 4.7, 4.8, 4.9

L'operatore Trident è supportato con le seguenti versioni:

- Anthos on-Prem (VMware) e anthos on Bare Metal 1.8, 1.9, 1.10
- Kubernetes 1.17 o versione successiva (ultimo: 1.23)
- OpenShift 4.7, 4.8, 4.9

> Gli utenti di Red Hat OpenShift Container Platform potrebbero notare che il file initiator name.iscsi è vuoto se si utilizza una versione inferiore a 4.6.8. Questo è un bug identificato da RedHat per essere corretto con OpenShift 4.6.8. Vedi questo "annuncio di risoluzione dei bug". NetApp consiglia di utilizzare Astra Trident su OpenShift 4.6.8 e versioni successive.

Astra Trident lavora anche con una serie di altre offerte Kubernetes completamente gestite e autogestite, tra cui Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service (AKS), Rancher e VMware Tanzu Portfolio.

Back-end supportati (storage)

Per utilizzare Astra Trident, sono necessari uno o più dei seguenti backend supportati:

- Amazon FSX per NetApp ONTAP
- Azure NetApp Files
- Archivio dati Astra
- Cloud Volumes ONTAP
- Cloud Volumes Service per GCP
- FAS/AFF/Select 9.3 o versione successiva
- Array All SAN (ASA) NetApp
- Software NetApp HCI/Element 11 o versione successiva

Requisiti delle funzionalità

La tabella seguente riassume le funzionalità disponibili con questa release di Astra Trident e le versioni di Kubernetes supportate.

Funzione	Versione di Kubernetes	Sono richiesti i gate delle funzionalità?
Trident CSI	1.17 e versioni successive	No
Snapshot dei volumi	1.17 e versioni successive	No
PVC dalle istantanee dei volumi	1.17 e versioni successive	No
Ridimensionamento di iSCSI PV	1.17 e versioni successive	No
CHAP bidirezionale ONTAP	1.17 e versioni successive	No
Policy di esportazione dinamiche	1.17 e versioni successive	No
Operatore Trident	1.17 e versioni successive	No
Preparazione automatica dei nodi di lavoro (beta)	1.17 e versioni successive	No
Topologia CSI	1.17 e versioni successive	No

Sistemi operativi host testati

Per impostazione predefinita, Astra Trident viene eseguito in un container e, di conseguenza, viene eseguito su qualsiasi worker Linux. Tuttavia, questi lavoratori devono essere in grado di montare i volumi forniti da Astra Trident utilizzando il client NFS standard o iSCSI Initiator, a seconda dei backend utilizzati.

Sebbene Astra Trident non supporti ufficialmente sistemi operativi specifici, le seguenti distribuzioni Linux sono note per funzionare:

- Versioni di RedHat CoreOS (RHCOS) supportate da OpenShift Container Platform
- RHEL o CentOS 7.4 o versione successiva
- Ubuntu 18.04 o versione successiva

Il `tridentctl` Utility può essere eseguita anche su una qualsiasi di queste distribuzioni di Linux.

Configurazione dell'host

A seconda del backend in uso, le utility NFS e/o iSCSI devono essere installate su tutti gli utenti del cluster. Vedere "qui" per ulteriori informazioni.

Configurazione del sistema storage

Astra Trident potrebbe richiedere alcune modifiche a un sistema storage prima che possa essere utilizzato da una configurazione di back-end. Vedere "qui" per ulteriori informazioni.

Immagini container e corrispondenti versioni di Kubernetes

Per le installazioni a gapping d'aria, l'elenco seguente è un riferimento alle immagini dei container necessarie per installare Astra Trident. Utilizzare `tridentctl images` per verificare l'elenco delle immagini container necessarie.

Versione di Kubernetes	Immagine container
v1.17.0	• netapp/tridente:22.01.1 • netapp/trident-autosupport:22.01 • k8s.gcr.io/sig-storage/csi-provisioner:v2.2.2 • k8s.gcr.io/sig-storage/csi-attacher:v3.4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0 • netapp/trident-operator:22.01.1 (opzionale)
v1.18.0	• netapp/tridente:22.01.1 • netapp/trident-autosupport:22.01 • k8s.gcr.io/sig-storage/csi-provisioner:v2.2.2 • k8s.gcr.io/sig-storage/csi-attacher:v3.4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0 • netapp/trident-operator:22.01.1 (opzionale)

Versione di Kubernetes	Immagine container
v1.19.0	• netapp/tridente:22.01.1 • netapp/trident-autosupport:22.01 • k8s.gcr.io/sig-storage/csi-provisioner:v2.2.2 • k8s.gcr.io/sig-storage/csi-attacher:v3.4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0 • netapp/trident-operator:22.01.1 (opzionale)
v1.20.0	• netapp/tridente:22.01.1 • netapp/trident-autosupport:22.01 • k8s.gcr.io/sig-storage/csi-provisioner:v3.1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3.4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0 • netapp/trident-operator:22.01.1 (opzionale)
v1.21.1.0	• netapp/tridente:22.01.1 • netapp/trident-autosupport:22.01 • k8s.gcr.io/sig-storage/csi-provisioner:v3.1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3.4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0 • netapp/trident-operator:22.01.1 (opzionale)

Versione di Kubernetes	Immagine container
v1.22.0	• netapp/tridente:22.01.1 • netapp/trident-autosupport:22.01 • k8s.gcr.io/sig-storage/csi-provisioner:v3.1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3.4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0 • netapp/trident-operator:22.01.1 (opzionale)
v1.23.0	• netapp/tridente:22.01.1 • netapp/trident-autosupport:22.01 • k8s.gcr.io/sig-storage/csi-provisioner:v3.1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3.4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0 • netapp/trident-operator:22.01.1 (opzionale)

Su Kubernetes versione 1.20 e successive, utilizzare il validato `k8s.gcr.io/sig-storage/csi-snapshotter:v4.x` immagine solo se `v1` la versione di sta servendo `volumesnapshots.snapshot.storage.k8s.io` CRD. Se il `v1beta1` La versione sta servendo il CRD con/senza `v1` versione, utilizzare il validato `k8s.gcr.io/sig-storage/csi-snapshotter:v3.x` immagine.

Panoramica dell'implementazione

Puoi implementare Astra Trident usando l'operatore Trident o con `tridentctl`.

Scegliere il metodo di implementazione

Per determinare quale metodo di implementazione utilizzare, considerare quanto segue:

Perché dovrei utilizzare l'operatore Trident?

Il "Operatore Trident" È un ottimo modo per gestire dinamicamente le risorse di Astra Trident e automatizzare la fase di setup. Alcuni prerequisiti devono essere soddisfatti. Vedere "i requisiti".

L'operatore Trident offre diversi vantaggi, come indicato di seguito.

Funzionalità di riparazione automatica

È possibile monitorare un'installazione di Astra Trident e prendere attivamente misure per risolvere problemi, ad esempio quando l'implementazione viene eliminata o se viene modificata accidentalmente. Quando l'operatore è impostato come implementazione, un `trident-operator-<generated-id>` pod creato. Questo pod associa un `TridentOrchestrator` CR con un'installazione Astra Trident e garantisce sempre che sia attivo un solo prodotto `TridentOrchestrator`. In altre parole, l'operatore garantisce che vi sia una sola istanza di Astra Trident nel cluster e ne controlla la configurazione, assicurandosi che l'installazione sia idempotente. Quando vengono apportate modifiche all'installazione (ad esempio, l'eliminazione dell'implementazione o del demonset di nodi), l'operatore li identifica e li corregge singolarmente.

Semplici aggiornamenti alle installazioni esistenti

È possibile aggiornare facilmente un'implementazione esistente con l'operatore. È sufficiente modificare `TridentOrchestrator` CR per aggiornare un'installazione. Ad esempio, si consideri uno scenario in cui è necessario abilitare Astra Trident per generare i log di debug.

A tale scopo, applicare una patch al `TridentOrchestrator` da impostare `spec.debug` a. `true`:

```
kubectl patch torc <trident-orchestrator-name> -n trident --type=merge -p
'{"spec":{"debug":true}}'
```

Dopo `TridentOrchestrator` viene aggiornato, l'operatore elabora gli aggiornamenti e le patch dell'installazione esistente. Questo potrebbe attivare la creazione di nuovi pod per modificare l'installazione di conseguenza.

Gestisce automaticamente gli aggiornamenti di Kubernetes

Quando la versione di Kubernetes del cluster viene aggiornata a una versione supportata, l'operatore aggiorna automaticamente un'installazione di Astra Trident esistente e la modifica per garantire che soddisfi i requisiti della versione di Kubernetes.

> Se il cluster viene aggiornato a una versione non supportata, l'operatore impedisce l'installazione di Astra Trident. Se Astra Trident è già stato installato con l'operatore, viene visualizzato un avviso per indicare che Astra Trident è installato su una versione di Kubernetes non supportata.

Perché dovrei usare Helm?

Se si utilizzano altre applicazioni che vengono gestite utilizzando Helm, a partire da Astra Trident 21.01, è possibile gestire la distribuzione anche utilizzando Helm.

Quando dovrei usare `tridenctl`?

Se si dispone di un'implementazione esistente che deve essere aggiornata o se si desidera personalizzare in modo efficace l'implementazione, è necessario esaminare l'utilizzo di "tridentctl". Questo è il metodo convenzionale per implementare Astra Trident.

Considerazioni sul passaggio da un metodo di implementazione all'altro

Non è difficile immaginare uno scenario in cui si desidera passare da un metodo di implementazione all'altro. Prima di tentare di spostarsi da un, tenere in considerazione quanto segue `tridentctl` implementazione su

un'implementazione basata su operatore o viceversa:

- Utilizzare sempre lo stesso metodo per disinstallare Astra Trident. Se hai implementato con `tridentctl`, utilizzare la versione appropriata di `tridentctl` Binario per disinstallare Astra Trident. Allo stesso modo, se si esegue la distribuzione con l'operatore, è necessario modificare `TridentOrchestrator` CR e set `spec.uninstall=true` Per disinstallare Astra Trident.
- Se si desidera rimuovere e utilizzare un'implementazione basata su operatore `tridentctl` Per implementare Astra Trident, devi prima modificarlo `TridentOrchestrator` e impostare `spec.uninstall=true` Per disinstallare Astra Trident. Quindi eliminare `TridentOrchestrator` e l'implementazione dell'operatore. È quindi possibile installare utilizzando `tridentctl`.
- Se si dispone di un'implementazione manuale basata su operatore e si desidera utilizzare l'implementazione dell'operatore Trident basata su Helm, è necessario prima disinstallare manualmente l'operatore, quindi eseguire l'installazione di Helm. Ciò consente a Helm di implementare l'operatore Trident con le etichette e le annotazioni richieste. In caso contrario, l'implementazione dell'operatore Trident basata su Helm avrà esito negativo, con un errore di convalida dell'etichetta e un errore di convalida dell'annotazione. Se si dispone di un `tridentctl`L'implementazione basata su consente di utilizzare l'implementazione basata su Helm senza problemi.

Comprendere le modalità di implementazione

Ci sono tre modi per implementare Astra Trident.

Implementazione standard

L'implementazione di Trident su un cluster Kubernetes comporta due operazioni da parte del programma di installazione di Astra Trident:

- Recupero delle immagini container tramite Internet
- Creazione di un demonset di implementazione e/o nodi, che consente di attivare i pod Astra Trident su tutti i nodi idonei nel cluster Kubernetes.

Un'implementazione standard come questa può essere eseguita in due modi diversi:

- Utilizzo di `tridentctl install`
- Utilizzando l'operatore Trident. È possibile implementare l'operatore Trident manualmente o utilizzando Helm.

Questa modalità di installazione è il modo più semplice per installare Astra Trident e funziona per la maggior parte degli ambienti che non impongono restrizioni di rete.

Implementazione offline

Per eseguire un'implementazione con aria compressa, è possibile utilizzare `--image-registry` contrassegno durante l'invocazione `tridentctl install` per puntare a un registro di immagini privato. Se si esegue l'implementazione con l'operatore Trident, è possibile specificare in alternativa `spec.imageRegistry` nel `TridentOrchestrator`. Questo registro deve contenere "Immagine di Trident", il "Immagine Trident AutoSupport"E le immagini sidecar CSI come richiesto dalla versione di Kubernetes.

Per personalizzare l'implementazione, è possibile utilizzare `tridentctl` Generare i manifesti per le risorse di Trident. Ciò include la distribuzione, il demonset, l'account del servizio e il ruolo del cluster creato da Astra Trident durante l'installazione.

Per ulteriori informazioni sulla personalizzazione della distribuzione, consultare i seguenti collegamenti:

- "Personalizza la tua implementazione basata su operatore"
  *

> Se si utilizza un repository di immagini privato, è necessario aggiungere `/k8scsi` Per le versioni di Kubernetes precedenti alla 1.17 o. `/sig-storage` Per le versioni di Kubernetes successive alla 1.17 fino alla fine dell'URL privato del Registro di sistema. Quando si utilizza un registro di sistema privato per `tridentctl` implementazione, è necessario utilizzare `--trident-image` e. `--autosupport-image` in combinazione con `--image-registry`. Se stai implementando Astra Trident utilizzando l'operatore Trident, assicurati che orchestrator CR includa `tridentImage` e. `autosupportImage` nei parametri di installazione.

Implementazione remota

Di seguito viene riportata una panoramica generale del processo di implementazione remota:

- Implementare la versione appropriata di `kubectl` Sul computer remoto da cui si desidera implementare Astra Trident.
- Copiare i file di configurazione dal cluster Kubernetes e impostare `KUBECONFIG` variabile di ambiente sul computer remoto.
- Avviare un `kubectl get nodes` Per verificare che sia possibile connettersi al cluster Kubernetes richiesto.
- Completare l'implementazione dal computer remoto utilizzando i passaggi di installazione standard.

Altre opzioni di configurazione note

Quando si installa Astra Trident sui prodotti del portfolio VMware Tanzu:

- Il cluster deve supportare workload con privilegi.
- Il `--kubelet-dir` flag deve essere impostato sulla posizione della directory di kubelet. Per impostazione predefinita, questo è `/var/vcap/data/kubelet`.

  Specificare la posizione del kubelet utilizzando `--kubelet-dir` È noto per lavorare con Trident Operator, Helm e. `tridentctl` implementazioni.

Implementazione con l'operatore Trident

Puoi implementare Astra Trident con l'operatore Trident. È possibile implementare l'operatore Trident manualmente o utilizzando Helm.

> Se non si è ancora familiarizzato con il "concetti di base", è il momento ideale per farlo.

Di cosa hai bisogno

Per implementare Astra Trident, devono essere soddisfatti i seguenti prerequisiti:

- Si dispone dei privilegi completi per un cluster Kubernetes supportato che esegue Kubernetes 1.17 e versioni successive.
- Hai accesso a un sistema storage NetApp supportato.

- È possibile montare volumi da tutti i nodi di lavoro Kubernetes.
- Hai un host Linux con `kubectl` (o. `oc`, Se si utilizza OpenShift) installato e configurato per gestire il cluster Kubernetes che si desidera utilizzare.
- È stato impostato il `KUBECONFIG` Variabile di ambiente che punta alla configurazione del cluster Kubernetes.
- È stata attivata la "Porte caratteristiche richieste da Astra Trident".
- Se utilizzi Kubernetes con Docker Enterprise, "Seguire la procedura per abilitare l'accesso CLI".

Hai tutto questo? Fantastico! Iniziamo.

Implementare l'operatore Trident utilizzando Helm

Eseguire i passaggi elencati per implementare l'operatore Trident utilizzando Helm.

Di cosa hai bisogno

Oltre ai prerequisiti elencati in precedenza, per implementare l'operatore Trident utilizzando Helm, è necessario disporre di quanto segue:

- Kubernetes 1.17 e versioni successive
- Helm versione 3

Fasi

1. Aggiungere il repository Helm di Trident:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. Utilizzare `helm install` e specificare un nome per la distribuzione. Vedere il seguente esempio:

```
helm install <release-name> netapp-trident/trident-operator --version
22.1.0 --namespace <trident-namespace>
```

> Se non è stato ancora creato uno spazio dei nomi per Trident, è possibile aggiungere `--create-namespace` al `helm install` comando. Helm crea automaticamente lo spazio dei nomi.

Esistono due modi per passare i dati di configurazione durante l'installazione:

- `--values` (o. `-f`): Specificare un file YAML con override. Questo valore può essere specificato più volte e il file più a destra avrà la precedenza.
- `--set`: Specificare le sostituzioni sulla riga di comando.

Ad esempio, per modificare il valore predefinito di `debug`, eseguire quanto segue `--set` comando:

```
$ helm install <name> netapp-trident/trident-operator --version 22.1.0
--set tridentDebug=true
```

Il `values.yaml` Il file, che fa parte del grafico Helm, fornisce l'elenco delle chiavi e i relativi valori predefiniti.

`helm list` mostra i dettagli dell'installazione, ad esempio nome, spazio dei nomi, grafico, stato, versione dell'applicazione, numero di revisione e così via.

Implementare l'operatore Trident manualmente

Eseguire i passaggi elencati per implementare manualmente l'operatore Trident.

Fase 1: Qualificare il cluster Kubernetes

La prima cosa da fare è accedere all'host Linux e verificare che stia gestendo un "Cluster Kubernetes supportato" disporre dei privilegi necessari per.

> Con OpenShift, utilizzare `oc` invece di `kubectl` in tutti gli esempi che seguono, accedere come **system:admin** eseguendo `oc login -u system:admin` oppure `oc login -u kube-admin`.

Per verificare se la versione di Kubernetes è successiva alla 1.17, eseguire il seguente comando:

```
kubectl version
```

Per verificare se si dispone dei privilegi di amministratore del cluster Kubernetes, eseguire il seguente comando:

```
kubectl auth can-i '*' '*' --all-namespaces
```

Per verificare se è possibile avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il sistema di storage sulla rete pod, eseguire il seguente comando:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Fase 2: Scaricare e configurare l'operatore

> A partire da 21.01, l'operatore Trident ha un ambito cluster. L'utilizzo dell'operatore Trident per installare Trident richiede la creazione di `TridentOrchestrator` Definizione personalizzata delle risorse (CRD) e definizione di altre risorse. Prima di installare Astra Trident, eseguire questa procedura per configurare l'operatore.

1. Scaricare l'ultima versione di "Pacchetto di installazione Trident" Dalla sezione *Downloads* ed estrarla.

```
wget https://github.com/NetApp/trident/releases/download/v21.04/trident-
installer-21.04.tar.gz
tar -xf trident-installer-21.04.tar.gz
cd trident-installer
```

2. Utilizzare il manifesto CRD appropriato per creare `TridentOrchestrator` CRD. Quindi, creare un `TridentOrchestrator` Custom Resource in seguito per creare un'installazione da parte dell'operatore.

   Eseguire il seguente comando:

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. Dopo il `TridentOrchestrator` Viene creato un CRD, creare le seguenti risorse necessarie per l'implementazione dell'operatore:

   - Un account di servizio per l'operatore
   - Un ClusterRole e ClusterRoleBinding al ServiceAccount
   - Una policy PodSecurityPolicy dedicata
   - L'operatore stesso

     Il programma di installazione di Trident contiene i manifesti per la definizione di queste risorse. Per impostazione predefinita, l'operatore viene implementato in `trident` namespace. Se il `trident` namespace non esiste, utilizzare il seguente manifesto per crearne uno.

```
$ kubectl apply -f deploy/namespace.yaml
```

4. Per implementare l'operatore in uno spazio dei nomi diverso da quello predefinito `trident` namespace, è necessario aggiornare `serviceaccount.yaml`, `clusterrolebinding.yaml` e. `operator.yaml` manifesta e genera il tuo `bundle.yaml`.

   Eseguire il comando seguente per aggiornare i manifesti YAML e generare il `bundle.yaml` utilizzando il `kustomization.yaml`:

```
kubectl kustomize deploy/ > deploy/bundle.yaml
```

   Eseguire il seguente comando per creare le risorse e implementare l'operatore:

```
kubectl create -f deploy/bundle.yaml
```

5. Per verificare lo stato dell'operatore dopo l'implementazione, procedere come segue:

```
$ kubectl get deployment -n <operator-namespace>
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m

$ kubectl get pods -n <operator-namespace>
NAME                                READY   STATUS             RESTARTS
AGE
trident-operator-54cb664d-lnjxh     1/1     Running            0
3m
```

L'implementazione dell'operatore crea correttamente un pod in esecuzione su uno dei nodi di lavoro nel cluster.

> In un cluster Kubernetes dovrebbe esserci solo **un'istanza** dell'operatore. Non creare implementazioni multiple dell'operatore Trident.

Fase 3: Creazione `TridentOrchestrator` E installare Trident

Ora sei pronto per installare Astra Trident usando l'operatore! Per questo è necessario creare `TridentOrchestrator`. Il programma di installazione di Trident include definizioni di esempio per la creazione `TridentOrchestrator`. In questo modo viene eseguita un'installazione in `trident` namespace.

```
$ kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

$ kubectl describe torc trident
Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:21.04
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Enable Node Prep:          false
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:21.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v21.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

L'operatore Trident consente di personalizzare il modo in cui Astra Trident viene installato utilizzando gli attributi in `TridentOrchestrator` spec. Vedere "Personalizza la tua implementazione Trident".

Lo Stato di `TridentOrchestrator` Indica se l'installazione ha avuto esito positivo e visualizza la versione di Trident installata.

Stato	Descrizione
Installazione in corso	L'operatore sta installando Astra Trident `TridentOrchestrator` CR.
Installato	Astra Trident è stato installato correttamente.
Disinstallazione in corso	L'operatore sta disinstallando Astra Trident, perché `spec.uninstall=true`.
Disinstallato	Astra Trident disinstallato.
Non riuscito	L'operatore non ha potuto installare, applicare patch, aggiornare o disinstallare Astra Trident; l'operatore tenterà automaticamente di eseguire il ripristino da questo stato. Se lo stato persiste, è necessario eseguire la risoluzione dei problemi.
Aggiornamento in corso	L'operatore sta aggiornando un'installazione esistente.
Errore	Il `TridentOrchestrator` non viene utilizzato. Un'altra esiste già.

Durante l'installazione, lo stato di `TridentOrchestrator` modifiche da `Installing` a. `Installed`. Se si osserva `Failed` e l'operatore non è in grado di eseguire il ripristino da solo, è necessario controllare i registri dell'operatore. Vedere "risoluzione dei problemi" sezione.

Puoi verificare se l'installazione di Astra Trident è stata completata dando un'occhiata ai pod creati:

```
$ kubectl get pod -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-7d466bf5c7-v4cpw        5/5     Running   0           1m
trident-csi-mr6zc                   2/2     Running   0           1m
trident-csi-xrp7w                   2/2     Running   0           1m
trident-csi-zh2jt                   2/2     Running   0           1m
trident-operator-766f7b8658-ldzsv   1/1     Running   0           3m
```

È anche possibile utilizzare `tridentctl` Per verificare la versione di Astra Trident installata.

```
$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.04.0        | 21.04.0        |
+----------------+----------------+
```

Ora puoi continuare a creare un back-end. Vedere "attività post-implementazione".

Per la risoluzione dei problemi durante l'implementazione, consultare "risoluzione dei problemi" sezione.

Personalizzare l'implementazione dell'operatore Trident

L'operatore Trident consente di personalizzare il modo in cui Astra Trident viene installato utilizzando gli attributi in `TridentOrchestrator` spec.

Per un elenco degli attributi, consultare la tabella seguente:

Parametro	Descrizione	Predefinito
`namespace`	Spazio dei nomi in cui installare Astra Trident	"predefinito"
`debug`	Attiva il debug per Astra Trident	falso
`IPv6`	Installare Astra Trident su IPv6	falso
`k8sTimeout`	Timeout per le operazioni Kubernetes	30 sec
`silenceAutosupport`	Non inviare pacchetti AutoSupport automaticamente a NetApp	falso
`enableNodePrep`	Gestire automaticamente le dipendenze dei nodi di lavoro (**BETA**)	falso
`autosupportImage`	L'immagine del contenitore per la telemetria AutoSupport	"netapp/trident-autosupport:21.04.0"
`autosupportProxy`	Indirizzo/porta di un proxy per l'invio di telemetria AutoSupport	"http://proxy.example.com:8888"
`uninstall`	Flag utilizzato per disinstallare Astra Trident	falso
`logFormat`	Formato di registrazione Astra Trident da utilizzare [text,json]	"testo"
`tridentImage`	Immagine Astra Trident da installare	"netapp/trident:21.04"
`imageRegistry`	Percorso al registro interno, del formato `<registry FQDN>[:port][/subpath]`	"k8s.gcr.io/sig-storage (k8s 1.17+) o quay.io/k8scsi"
`kubeletDir`	Percorso della directory del kubelet sull'host	"/var/lib/kubelet"
`wipeout`	Un elenco di risorse da eliminare per eseguire una rimozione completa di Astra Trident	

Parametro	Descrizione	Predefinito
`imagePullSecrets`	Secrets (segreti) per estrarre immagini da un registro interno	
`controllerPluginNodeSelector`	Selettori di nodi aggiuntivi per i pod che eseguono il Plugin CSI del controller Trident. Segue lo stesso formato di pod.spec.nodeSelector.	Nessuna impostazione predefinita; opzionale
`controllerPluginTolerations`	Ignora le tolleranze per i pod che eseguono il Plugin CSI del controller Trident. Segue lo stesso formato di pod.spec.Tolerations.	Nessuna impostazione predefinita; opzionale
`nodePluginNodeSelector`	Selettori di nodi aggiuntivi per i pod che eseguono il Plugin CSI di Trident Node. Segue lo stesso formato di pod.spec.nodeSelector.	Nessuna impostazione predefinita; opzionale
`nodePluginTolerations`	Ignora le tolleranze per i pod che eseguono il Plugin CSI di Trident Node. Segue lo stesso formato di pod.spec.Tolerations.	Nessuna impostazione predefinita; opzionale

`spec.namespace` è specificato in `TridentOrchestrator` Per indicare in quale spazio dei nomi Astra Trident è installato. Questo parametro **non può essere aggiornato dopo l'installazione di Astra Trident**. Il tentativo di eseguire questa operazione causa lo stato di `TridentOrchestrator` per passare a. `Failed`. Astra Trident non deve essere migrato tra spazi dei nomi.

La preparazione automatica dei nodi di lavoro è una funzionalità * beta* che deve essere utilizzata solo in ambienti non di produzione.

Per ulteriori informazioni sulla formattazione dei parametri del pod, vedere "Assegnazione di pod ai nodi".

È possibile utilizzare gli attributi menzionati in precedenza per la definizione `TridentOrchestrator` per personalizzare l'installazione. Ecco un esempio:

```
$ cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
```

Ecco un altro esempio che mostra come Trident può essere implementato con i selettori di nodo:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  controllerPluginNodeSelector:
    nodetype: master
  nodePluginNodeSelector:
    storage: netapp
```

Se si desidera personalizzare l'installazione oltre ciò che si desidera `TridentOrchestrator` gli argomenti lo consentono, dovresti considerare di utilizzare `tridentctl` Per generare manifesti YAML personalizzati che è possibile modificare in base alle esigenze.

Implementare con tridentctl

Puoi implementare Astra Trident utilizzando `tridentctl`.

Se non si è ancora familiarizzato con il "concetti di base", è il momento ideale per farlo.

Per personalizzare l'implementazione, vedere "qui".

Di cosa hai bisogno

Per implementare Astra Trident, devono essere soddisfatti i seguenti prerequisiti:

- Si dispone dei privilegi completi per un cluster Kubernetes supportato.
- Hai accesso a un sistema storage NetApp supportato.
- È possibile montare volumi da tutti i nodi di lavoro Kubernetes.
- Hai un host Linux con `kubectl` (o. `oc`, Se si utilizza OpenShift) installato e configurato per gestire il cluster Kubernetes che si desidera utilizzare.
- È stato impostato il `KUBECONFIG` Variabile di ambiente che punta alla configurazione del cluster Kubernetes.
- È stata attivata la "Porte caratteristiche richieste da Astra Trident".
- Se utilizzi Kubernetes con Docker Enterprise, "Seguire la procedura per abilitare l'accesso CLI".

Hai tutto questo? Fantastico! Iniziamo.

Per informazioni sulla personalizzazione della distribuzione, vedere "qui".

Fase 1: Qualificare il cluster Kubernetes

La prima cosa da fare è accedere all'host Linux e verificare che stia gestendo un "Cluster Kubernetes supportato" disporre dei privilegi necessari per.

> Con OpenShift, si utilizza `oc` invece di `kubectl` in tutti gli esempi riportati di seguito, eseguire prima l'accesso come **system:admin** `oc login -u system:admin` oppure `oc login -u kube-admin`.

Per controllare la versione di Kubernetes, eseguire il seguente comando:

```
kubectl version
```

Per verificare se si dispone dei privilegi di amministratore del cluster Kubernetes, eseguire il seguente comando:

```
kubectl auth can-i '*' '*' --all-namespaces
```

Per verificare se è possibile avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il sistema di storage sulla rete pod, eseguire il seguente comando:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Identificare la versione del server Kubernetes. Lo userai quando installi Astra Trident.

Fase 2: Scaricare ed estrarre il programma di installazione

> Il programma di installazione di Trident crea un pod Trident, configura gli oggetti CRD utilizzati per mantenere il proprio stato e inizializza i sidecar CSI che eseguono azioni, come il provisioning e il collegamento di volumi agli host del cluster.

È possibile scaricare l'ultima versione di "Pacchetto di installazione Trident" Dalla sezione *Downloads* ed estrarla.

Ad esempio, se la versione più recente è 21.07.1:

```
wget https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
tar -xf trident-installer-21.07.1.tar.gz
cd trident-installer
```

Fase 3: Installare Astra Trident

Installare Astra Trident nello spazio dei nomi desiderato eseguendo `tridentctl install` comando.

```
$ ./tridentctl install -n trident
....
INFO Starting Trident installation.                namespace=trident
INFO Created service account.
INFO Created cluster role.
INFO Created cluster role binding.
INFO Added finalizers to custom resource definitions.
INFO Created Trident service.
INFO Created Trident secret.
INFO Created Trident deployment.
INFO Created Trident daemonset.
INFO Waiting for Trident pod to start.
INFO Trident pod started.                          namespace=trident
pod=trident-csi-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=21.07.1
INFO Trident installation succeeded.
....
```

Al termine del programma di installazione, il suo aspetto sarà simile a questo. A seconda del numero di nodi nel cluster Kubernetes, è possibile osservare più pod:

```
$ kubectl get pod -n trident
NAME                           READY   STATUS    RESTARTS   AGE
trident-csi-679648bd45-cv2mx   4/4     Running   0          5m29s
trident-csi-vgc8n              2/2     Running   0          5m29s

$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.1        | 21.07.1        |
+----------------+----------------+
```

Se l'output è simile all'esempio precedente, questo passaggio è stato completato, ma Astra Trident non è ancora completamente configurato. Andare avanti e passare alla fase successiva. Vedere "attività post-implementazione".

Tuttavia, se il programma di installazione non viene completato correttamente o non viene visualizzato il valore **in esecuzione** `trident-csi-<generated id>`, la piattaforma non è stata installata.

Per la risoluzione dei problemi durante l'implementazione, consultare "risoluzione dei problemi" sezione.

Personalizzare l'implementazione tridentctl

Il programma di installazione di Trident consente di personalizzare gli attributi. Ad esempio, se l'immagine Trident è stata copiata in un repository privato, è possibile specificare il nome dell'immagine utilizzando `--trident-image`. Se l'immagine Trident e le immagini sidecar CSI necessarie sono state copiate in un repository privato, potrebbe essere preferibile specificare la posizione di tale repository utilizzando `--image-registry` switch, che assume la forma `<registry FQDN>[:port]`.

Per fare in modo che Astra Trident configuri automaticamente i nodi di lavoro, utilizzare `--enable-node-prep`. Per ulteriori informazioni sul funzionamento, vedere "qui".

> La preparazione automatica del nodo di lavoro è una funzionalità * beta* destinata all'utilizzo solo in ambienti non di produzione.

Se stai usando una distribuzione di Kubernetes, dove `kubelet` mantiene i dati su un percorso diverso dal solito `/var/lib/kubelet`, è possibile specificare il percorso alternativo utilizzando `--kubelet-dir`.

Se è necessario personalizzare l'installazione oltre a quanto consentito dall'argomento del programma di installazione, è possibile personalizzare i file di distribuzione. Utilizzando il `--generate-custom-yaml` Il parametro crea i seguenti file YAML nel programma di installazione `setup` directory:

- `trident-clusterrolebinding.yaml`
- `trident-deployment.yaml`
- `trident-crds.yaml`
- `trident-clusterrole.yaml`
- `trident-daemonset.yaml`
- `trident-service.yaml`
- `trident-namespace.yaml`
- `trident-serviceaccount.yaml`

Dopo aver generato questi file, è possibile modificarli in base alle proprie esigenze e utilizzarli `--use-custom-yaml` per installare l'implementazione personalizzata.

```
./tridentctl install -n trident --use-custom-yaml
```

Cosa succederà?

Dopo aver implementato Astra Trident, è possibile procedere con la creazione di un backend, la creazione di una classe di storage, il provisioning di un volume e il montaggio del volume in un pod.

Fase 1: Creazione di un backend

È ora possibile creare un backend che verrà utilizzato da Astra Trident per il provisioning dei volumi. A tale scopo, creare un `backend.json` che contiene i parametri necessari. I file di configurazione di esempio per diversi tipi di backend sono disponibili in `sample-input` directory.

Vedere "qui" per ulteriori informazioni su come configurare il file per il tipo di backend.

```
cp sample-input/<backend template>.json backend.json
vi backend.json
```

```
./tridentctl -n trident create backend -f backend.json
+-------------+----------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+-------------+----------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+----------------+--------------------------------------
+--------+---------+
```

Se la creazione non riesce, si è verificato un errore nella configurazione del back-end. È possibile visualizzare i log per determinare la causa eseguendo il seguente comando:

```
./tridentctl -n trident logs
```

Dopo aver risolto il problema, tornare all'inizio di questo passaggio e riprovare. Per ulteriori suggerimenti sulla risoluzione dei problemi, vedere "la risoluzione dei problemi" sezione.

Fase 2: Creazione di una classe di storage

Kubernetes consente agli utenti di eseguire il provisioning dei volumi utilizzando le dichiarazioni di volumi persistenti (PVC) che specificano a. "classe di storage" per nome. I dettagli sono nascosti agli utenti, ma una classe di storage identifica il provisioning utilizzato per tale classe (in questo caso Trident) e il significato di tale classe per il provisioning.

Creare una classe di storage Kubernetes gli utenti specificheranno quando desiderano un volume. La configurazione della classe deve modellare il backend creato nel passaggio precedente, in modo che Astra Trident lo utilizzi per il provisioning di nuovi volumi.

La classe di storage più semplice da utilizzare è basata su `sample-input/storage-class-csi.yaml.templ` file fornito con il programma di installazione, in sostituzione *`BACKEND_TYPE`* con il nome del driver di storage.

```
./tridentctl -n trident get backend
+-------------+----------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+----------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+----------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

Si tratta di un oggetto Kubernetes, quindi si utilizza `kubectl` Per crearlo in Kubernetes.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

Ora dovrebbe essere visualizzata una classe di storage **Basic-csi** in Kubernetes e Astra Trident, mentre Astra Trident avrebbe scoperto i pool sul backend.

```
kubectl get sc basic-csi
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

Fase 3: Eseguire il provisioning del primo volume

Ora sei pronto per eseguire il provisioning dinamico del tuo primo volume. Per eseguire questa operazione, creare un Kubernetes "richiesta di volume persistente" (PVC).

Creare un PVC per un volume che utilizzi la classe di storage appena creata.

Vedere `sample-input/pvc-basic-csi.yaml` ad esempio. Assicurarsi che il nome della classe di storage corrisponda a quello creato.

```
kubectl create -f sample-input/pvc-basic-csi.yaml

kubectl get pvc --watch
NAME      STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS   AGE
basic     Pending
basic          1s
basic     Pending   pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   0
basic          5s
basic     Bound     pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   1Gi
RWO           basic          7s
```

Fase 4: Montare i volumi in un pod

Ora montiamo il volume. Lanceremo un pod nginx che monta il PV sotto `/usr/share/nginx/html`.

```
cat << EOF > task-pv-pod.yaml
kind: Pod
apiVersion: v1
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
EOF
kubectl create -f task-pv-pod.yaml
```

```
# Wait for the pod to start
kubectl get pod --watch

# Verify that the volume is mounted on /usr/share/nginx/html
kubectl exec -it task-pv-pod -- df -h /usr/share/nginx/html

# Delete the pod
kubectl delete pod task-pv-pod
```

A questo punto, il pod (applicazione) non esiste più, ma il volume è ancora presente. Se lo si desidera, è possibile utilizzarlo da un altro pod.

Per eliminare il volume, eliminare la richiesta di rimborso:

```
kubectl delete pvc basic
```

È ora possibile eseguire attività aggiuntive, come ad esempio:

- "Configurare backend aggiuntivi."
- "Creare ulteriori classi di storage."

Gestire Astra Trident

Aggiorna Astra Trident

Astra Trident segue una cadenza di rilascio trimestrale, fornendo quattro release principali ogni anno di calendario. Ogni nuova release si basa sulle release precedenti, fornendo nuove funzionalità e miglioramenti delle performance, oltre a correzioni di bug e miglioramenti. Ti consigliamo di eseguire l'upgrade almeno una volta all'anno per sfruttare le nuove funzionalità di Astra Trident.

> L'aggiornamento a una release con cinque release in anticipo richiede l'esecuzione di un aggiornamento a più fasi.

Determinare la versione a cui eseguire l'aggiornamento

- È possibile eseguire l'aggiornamento a. `YY.MM` rilasciare dal `YY-1.MM` release ed eventuali release intermedie. Ad esempio, è possibile eseguire un aggiornamento diretto alla versione 20.07 dalla versione 19.07 e successive (incluse le versioni DOT, ad esempio 19.07.1).
- Se si dispone di una versione precedente, è necessario eseguire un aggiornamento a più fasi. Ciò richiede l'aggiornamento alla versione più recente che si adatta alla finestra di quattro release. Ad esempio, se si utilizza 18.07 e si desidera eseguire l'aggiornamento alla versione 20.07, seguire la procedura di aggiornamento in più fasi come indicato di seguito:
  - Primo aggiornamento da 18.07 a 19.07. Consultare la documentazione della rispettiva release per ottenere istruzioni specifiche per l'aggiornamento.
  - Eseguire l'aggiornamento da 19.07 a 20.07.

> Tutti gli aggiornamenti per le versioni 19.04 e precedenti richiedono la migrazione dei metadati di Astra Trident da soli `etcd` Agli oggetti CRD. Verificare la documentazione della release per comprendere il funzionamento dell'aggiornamento.

> Durante l'aggiornamento, è importante fornire `parameter.fsType` poll `StorageClasses` Utilizzato da Astra Trident. È possibile eliminare e ricreare `StorageClasses` senza interrompere i volumi preesistenti. Si tratta di un **requisito** per l'applicazione dei contesti security per i volumi SAN. La directory sample input contiene esempi, come `storage-class-basic.yaml.templ` e `storage-class-bronze-default.yaml`.
> Per ulteriori informazioni, vedere "Problemi noti".

Quale percorso di upgrade dovrei scegliere?

È possibile eseguire l'aggiornamento utilizzando uno dei seguenti percorsi:

- Utilizzando l'operatore Trident.
- Utilizzo di `tridentctl`.

> CSI Volume Snapshots è ora una funzionalità GA, a partire da Kubernetes 1.20. Quando si aggiorna Astra Trident, tutti i CRS e i CRD di snapshot alfa precedenti (classi di snapshot dei volumi, snapshot dei volumi e contenuti di snapshot dei volumi) devono essere rimossi prima di eseguire l'aggiornamento. Fare riferimento a. "questo blog" Comprendere i passaggi necessari per la migrazione delle snapshot Alpha alle specifiche beta/GA.

È possibile utilizzare l'operatore Trident per eseguire l'aggiornamento se vengono soddisfatte le seguenti condizioni:

- Si utilizza CSI Trident (19.07 e versioni successive).
- Hai una release di Trident basata su CRD (19.07 e versioni successive).
- Non si sta eseguendo un'installazione personalizzata (utilizzando YAML personalizzati).

> Non utilizzare l'operatore per aggiornare Trident se si utilizza un `etcd`Versione Trident basata su (19.04 o precedente).

Se non si desidera utilizzare l'operatore o si dispone di un'installazione personalizzata che non può essere supportata dall'operatore, è possibile eseguire l'aggiornamento utilizzando `tridentctl`. Questo è il metodo preferito per gli aggiornamenti di Trident release 19.04 e precedenti.

Modifiche all'operatore

La release 21.01 di Astra Trident introduce alcune importanti modifiche architetturali all'operatore, vale a dire quanto segue:

- L'operatore è ora **con ambito cluster**. Le istanze precedenti dell'operatore Trident (versioni da 20.04 a 20.10) erano **namespace-scope**. Un operatore con ambito cluster è vantaggioso per i seguenti motivi:
  - Responsabilità delle risorse: L'operatore gestisce ora le risorse associate a un'installazione di Astra Trident a livello di cluster. Nell'ambito dell'installazione di Astra Trident, l'operatore crea e gestisce diverse risorse utilizzando `ownerReferences`. Manutenzione `ownerReferences` Su risorse con ambito cluster possono generare errori su alcuni distributori Kubernetes come OpenShift. Questo è mitigato da un operatore con ambito cluster. Per la riparazione automatica e l'applicazione di patch alle risorse Trident, questo è un requisito essenziale.
  - Pulizia durante la disinstallazione: Una rimozione completa di Astra Trident richiederebbe l'eliminazione di tutte le risorse associate. Un operatore con ambito spazio dei nomi potrebbe riscontrare problemi con la rimozione delle risorse con ambito del cluster (come ClusterRole, ClusterRoleBinding e PodSecurityPolicy) e portare a una pulizia incompleta. Un operatore con ambito cluster elimina questo problema. Gli utenti possono disinstallare completamente Astra Trident e installare di nuovo, se necessario.
- `TridentProvisioner` viene ora sostituito con `TridentOrchestrator` Come risorsa personalizzata utilizzata per installare e gestire Astra Trident. Inoltre, viene introdotto un nuovo campo in `TridentOrchestrator` spec. Gli utenti possono specificare che lo spazio dei nomi Trident deve essere installato/aggiornato utilizzando `spec.namespace` campo. Puoi dare un'occhiata a un esempio "qui".

Trova ulteriori informazioni

- "Eseguire l'aggiornamento utilizzando l'operatore Trident"
  *

Eseguire l'upgrade con l'operatore

È possibile aggiornare facilmente un'installazione Astra Trident esistente utilizzando l'operatore.

Di cosa hai bisogno

Per eseguire l'aggiornamento utilizzando l'operatore, devono essere soddisfatte le seguenti condizioni:

- È necessario disporre di un'installazione Astra Trident basata su CSI. Per verificare se CSI Trident è in esecuzione, esaminare i pod nello spazio dei nomi Trident. Se seguono `trident-csi-*` Schema di denominazione, si sta eseguendo CSI Trident.
- È necessario disporre di un'installazione di Trident basata su CRD. Questo rappresenta tutte le release della versione 19.07 e successive. Se si dispone di un'installazione basata su CSI, è molto probabile che si disponga di un'installazione basata su CRD.
- Se CSI Trident è stato disinstallato e i metadati dell'installazione persistono, è possibile eseguire l'aggiornamento utilizzando l'operatore.
- Deve esistere una sola installazione Astra Trident in tutti gli spazi dei nomi di un determinato cluster Kubernetes.
- Si dovrebbe utilizzare un cluster Kubernetes in esecuzione "versione 1.17 e successive".
- Se sono presenti CRD Alpha Snapshot, rimuoverli con `tridentctl obliviate alpha-snapshot-crd`. In questo modo vengono eliminati i CRD per le specifiche di snapshot alfa. Per gli snapshot esistenti che devono essere cancellati/migrati, vedere "questo blog".

> Quando si aggiorna Trident utilizzando l'operatore su OpenShift Container Platform, è necessario eseguire l'aggiornamento a Trident 21.01.1 o versione successiva. L'operatore Trident rilasciato con 21.01.0 contiene un problema noto che è stato risolto nel 21.01.1. Per ulteriori informazioni, vedere "Dettagli del problema su GitHub".

Aggiornare un'installazione di un operatore con ambito cluster

Per eseguire l'aggiornamento da **Trident 21.01 e versioni successive**, ecco la procedura da seguire.

Fasi

1. Eliminare l'operatore Trident utilizzato per installare l'istanza corrente di Astra Trident. Ad esempio, se si esegue l'aggiornamento da 21.01, eseguire il seguente comando:

```
kubectl delete -f 21.01/trident-installer/deploy/bundle.yaml -n trident
```

2. (Facoltativo) se si desidera modificare i parametri di installazione, modificare `TridentOrchestrator` Oggetto creato durante l'installazione di Trident. Ciò può includere modifiche, ad esempio la modifica dell'immagine Trident personalizzata, il Registro di sistema di immagini private da cui estrarre le immagini container, l'attivazione dei registri di debug o la specifica dei segreti di pull delle immagini.
3. Installare Astra Trident utilizzando `bundle.yaml` File che imposta l'operatore Trident per la nuova versione. Eseguire il seguente comando:

```
kubectl create -f 21.10.0/trident-installer/deploy/bundle.yaml -n
trident
```

Nell'ambito di questa fase, l'operatore 21.10.0 Trident identificherà un'installazione Astra Trident esistente e la aggiornerà alla stessa versione dell'operatore.

Aggiornare un'installazione dell'operatore con ambito namespace

Per eseguire l'aggiornamento da un'istanza di Astra Trident installata utilizzando l'operatore namespace-scoped (versioni da 20.07 a 20.10), ecco la serie di passaggi da seguire:

Fasi

1. Verificare lo stato dell'installazione Trident esistente. Per eseguire questa operazione, selezionare il valore **Status** di `TridentProvisioner`. Lo stato deve essere `Installed`.

```
$ kubectl describe tprov trident -n trident | grep Message: -A 3
Message:  Trident installed
Status:   Installed
Version:  v20.10.1
```

> Se viene visualizzato lo stato `Updating`, assicurarsi di risolverlo prima di procedere. Per un elenco dei possibili valori di stato, vedere "qui".

2. Creare il `TridentOrchestrator` CRD utilizzando il manifesto fornito con il programma di installazione di Trident.

```
# Download the release required [21.01]
$ mkdir 21.07.1
$ cd 21.07.1
$ wget
https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
$ tar -xf trident-installer-21.07.1.tar.gz
$ cd trident-installer
$ kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. Eliminare l'operatore con ambito dello spazio dei nomi utilizzando il relativo manifesto. Per completare questo passaggio, è necessario il `bundle.yaml` file utilizzato per implementare l'operatore con ambito dello spazio dei nomi. È possibile ottenere `bundle.yaml` dal "Repository di Trident". Assicurarsi di utilizzare la filiale appropriata.

> È necessario apportare le modifiche necessarie ai parametri di installazione di Trident (ad esempio, modificando i valori per `tridentImage`, `autosupportImage`, repository di immagini privato e fornitura `imagePullSecrets`) dopo aver eliminato l'operatore con ambito dello spazio dei nomi e prima di installare l'operatore con ambito del cluster. Per un elenco completo dei parametri che è possibile aggiornare, vedere "elenco dei parametri".

```
#Ensure you are in the right directory
$ pwd
$ /root/20.10.1/trident-installer

#Delete the namespace-scoped operator
$ kubectl delete -f deploy/bundle.yaml
serviceaccount "trident-operator" deleted
clusterrole.rbac.authorization.k8s.io "trident-operator" deleted
clusterrolebinding.rbac.authorization.k8s.io "trident-operator" deleted
deployment.apps "trident-operator" deleted
podsecuritypolicy.policy "tridentoperatorpods" deleted

#Confirm the Trident operator was removed
$ kubectl get all -n trident
NAME                                READY   STATUS    RESTARTS   AGE
pod/trident-csi-68d979fb85-dsrmn   6/6     Running   12         99d
pod/trident-csi-8jfhf              2/2     Running   6          105d
pod/trident-csi-jtnjz              2/2     Running   6          105d
pod/trident-csi-lcxvh              2/2     Running   8          105d

NAME                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
service/trident-csi   ClusterIP   10.108.174.125   <none>
34571/TCP,9220/TCP   105d

NAME                         DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR                                       AGE
daemonset.apps/trident-csi   3         3         3       3            3
kubernetes.io/arch=amd64,kubernetes.io/os=linux   105d

NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/trident-csi   1/1     1            1           105d

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/trident-csi-68d979fb85   1         1         1
105d
```

A questo punto, il `trident-operator-xxxxxxxxxx-xxxxx` pod eliminato.

4. (Facoltativo) se è necessario modificare i parametri di installazione, aggiornare `TridentProvisioner` spec. Tali modifiche potrebbero essere apportate, ad esempio, alla modifica del Registro di sistema dell'immagine privata per estrarre le immagini container, abilitare i registri di debug o specificare i segreti di pull delle immagini.

```
$  kubectl patch tprov <trident-provisioner-name> -n <trident-namespace>
--type=merge -p '{"spec":{"debug":true}}'
```

5. Installare l'operatore cluster-scoped.

> L'installazione dell'operatore con ambito cluster avvia la migrazione di `TridentProvisioner` oggetti a. `TridentOrchestrator` oggetti, elimina `TridentProvisioner` oggetti e il `tridentprovisioner` CRD e aggiorna Astra Trident alla versione dell'operatore cluster-scoped in uso. Nell'esempio seguente, Trident viene aggiornato alla versione 21.07.1.

> L'aggiornamento di Astra Trident utilizzando l'operatore con ambito cluster comporta la migrazione di `tridentProvisioner` a un `tridentOrchestrator` oggetto con lo stesso nome. Questo viene gestito automaticamente dall'operatore. Nell'aggiornamento verrà installato anche Astra Trident nello stesso namespace di prima.

```
#Ensure you are in the correct directory
$ pwd
$ /root/21.07.1/trident-installer

#Install the cluster-scoped operator in the **same namespace**
$ kubectl create -f deploy/bundle.yaml
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#All tridentProvisioners will be removed, including the CRD itself
$ kubectl get tprov -n trident
Error from server (NotFound): Unable to list "trident.netapp.io/v1,
Resource=tridentprovisioners": the server could not find the requested
resource (get tridentprovisioners.trident.netapp.io)

#tridentProvisioners are replaced by tridentOrchestrator
$ kubectl get torc
NAME      AGE
trident   13s

#Examine Trident pods in the namespace
$ kubectl get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m41s
trident-csi-xrst8                   2/2     Running   0          1m41s
trident-operator-5574dbbc68-nthjv   1/1     Running   0          1m52s

#Confirm Trident has been updated to the desired version
$ kubectl describe torc trident | grep Message -A 3
Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v21.07.1
```

Aggiornare un’installazione basata su Helm

Per aggiornare un’installazione basata su Helm, procedere come segue.

Fasi

1. Scarica l’ultima release di Astra Trident.
2. Utilizzare `helm upgrade` comando. Vedere il seguente esempio:

```
$ helm upgrade <name> trident-operator-21.07.1.tgz
```

dove `trident-operator-21.07.1.tgz` indica la versione alla quale si desidera eseguire l'aggiornamento.

3. Eseguire `helm list` per verificare che la versione del grafico e dell'applicazione sia stata aggiornata.

> Per passare i dati di configurazione durante l'aggiornamento, utilizzare `--set`.

Ad esempio, per modificare il valore predefinito di `tridentDebug`, eseguire il seguente comando:

```
$ helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
tridentDebug=true
```

Se corri `$ tridentctl logs`, vengono visualizzati i messaggi di debug.

> Se si impostano opzioni non predefinite durante l'installazione iniziale, assicurarsi che le opzioni siano incluse nel comando di aggiornamento, altrimenti i valori verranno ripristinati ai valori predefiniti.

Aggiornamento da un'installazione non eseguita dall'operatore

Se si dispone di un'istanza di CSI Trident che soddisfa i prerequisiti elencati in precedenza, è possibile eseguire l'aggiornamento all'ultima versione dell'operatore Trident.

Fasi

1. Scarica l'ultima release di Astra Trident.

```
# Download the release required [21.07.1]
$ mkdir 21.07.1
$ cd 21.07.1
$ wget
https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
$ tar -xf trident-installer-21.07.1.tar.gz
$ cd trident-installer
```

2. Creare il `tridentorchestrator` CRD dal manifesto.

```
$ kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. Implementare l'operatore.

```
#Install the cluster-scoped operator in the **same namespace**
$ kubectl create -f deploy/bundle.yaml
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          150d
trident-csi-xrst8                   2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv   1/1     Running   0          1m30s
```

4. Creare un `TridentOrchestrator` CR per l'installazione di Astra Trident.

```
#Create a tridentOrchestrator to initate a Trident install
$ cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident

$ kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s

#Confirm Trident was upgraded to the desired version
$ kubectl describe torc trident | grep Message -A 3
Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v21.07.1
```

I backend e i PVC esistenti sono automaticamente disponibili.

Upgrade con tridentctl

È possibile aggiornare facilmente un'installazione Astra Trident utilizzando `tridentctl`.

Considerazioni

Quando si esegue l'aggiornamento all'ultima release di Astra Trident, considerare quanto segue:

- A partire da Trident 20.01, solo la versione beta di "snapshot dei volumi" è supportato. Gli amministratori di Kubernetes devono fare attenzione a eseguire il backup o la conversione degli oggetti snapshot alfa in versione beta in modo sicuro per conservare le snapshot alfa legacy.
- La versione beta delle snapshot dei volumi introduce un set modificato di CRD e un controller di snapshot, entrambi da configurare prima di installare Astra Trident.

> "Questo blog" vengono illustrate le fasi della migrazione delle snapshot dei volumi alpha al formato beta.

A proposito di questa attività

La disinstallazione e la reinstallazione di Astra Trident funge da aggiornamento. Quando si disinstalla Trident, i PVC (Persistent Volume Claim) e PV (Persistent Volume) utilizzati dall'implementazione di Astra Trident non vengono cancellati. I PVS già forniti resteranno disponibili mentre Astra Trident è offline e Astra Trident effettuerà il provisioning dei volumi per i PVC creati nel frattempo una volta tornati online.

> Durante l'aggiornamento di Astra Trident, non interrompere il processo di aggiornamento. Assicurarsi che il programma di installazione venga completato.

Passi successivi dopo l'aggiornamento

Per utilizzare l'insieme completo di funzionalità disponibili nelle versioni più recenti di Trident (ad esempio, le snapshot dei volumi on-Demand), è possibile aggiornare i volumi utilizzando `tridentctl upgrade` comando.

Se sono presenti volumi legacy, è necessario aggiornarli da un tipo NFS/iSCSI al tipo CSI per poter utilizzare il set completo di nuove funzionalità di Astra Trident. Un PV legacy che è stato fornito da Trident supporta il set tradizionale di funzionalità.

Quando si decide di aggiornare i volumi al tipo CSI, considerare quanto segue:

- Potrebbe non essere necessario aggiornare tutti i volumi. I volumi creati in precedenza continueranno ad essere accessibili e funzioneranno normalmente.
- Un PV può essere montato come parte di un'implementazione/StatefulSet durante l'aggiornamento. Non è necessario mettere fuori servizio il deployment/StatefulSet.
- Non è possibile collegare un PV a un pod standalone durante l'aggiornamento. Chiudere il pod prima di aggiornare il volume.
- È possibile aggiornare solo un volume associato a un PVC. I volumi che non sono associati a PVC devono essere rimossi e importati prima dell'aggiornamento.

Esempio di aggiornamento del volume

Ecco un esempio che mostra come viene eseguito un aggiornamento di un volume.

1. Eseguire `kubectl get pv` Per elencare il PVS.

```
$ kubectl get pv
NAME                          CAPACITY     ACCESS MODES   RECLAIM POLICY
STATUS   CLAIM                   STORAGECLASS    REASON   AGE
default-pvc-1-a8475           1073741824   RWO            Delete
Bound    default/pvc-1           standard                19h
default-pvc-2-a8486           1073741824   RWO            Delete
Bound    default/pvc-2           standard                19h
default-pvc-3-a849e           1073741824   RWO            Delete
Bound    default/pvc-3           standard                19h
default-pvc-4-a84de           1073741824   RWO            Delete
Bound    default/pvc-4           standard                19h
trident                       2Gi          RWO            Retain
Bound    trident/trident                                  19h
```

Attualmente sono disponibili quattro PVS creati da Trident 20.07, utilizzando `netapp.io/trident` provisioner.

2. Eseguire `kubectl describe pv` Per ottenere i dettagli del PV.

```
$ kubectl describe pv default-pvc-2-a8486

Name:            default-pvc-2-a8486
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: netapp.io/trident
                 volume.beta.kubernetes.io/storage-class: standard
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    standard
Status:          Bound
Claim:           default/pvc-2
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        1073741824
Node Affinity:   <none>
Message:
Source:
    Type:      NFS (an NFS mount that lasts the lifetime of a pod)
    Server:    10.xx.xx.xx
    Path:      /trid_1907_alpha_default_pvc_2_a8486
    ReadOnly:  false
```

Il PV è stato creato utilizzando `netapp.io/trident` provisioner ed è del tipo NFS. Per supportare tutte le nuove funzioni fornite da Astra Trident, questo PV deve essere aggiornato al tipo CSI.

3. Eseguire `tridenctl upgrade volume <name-of-trident-volume>` Comando per aggiornare un volume Astra Trident legacy alla specifica CSI.

```
$ ./tridentctl get volumes -n trident
+----------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
|            NAME      |  SIZE   | STORAGE CLASS | PROTOCOL |
BACKEND UUID             | STATE  | MANAGED |
+----------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-3-a849e | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-1-a8475 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-4-a84de | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
+----------------------+---------+---------------+----------
+--------------------------------------+--------+---------+

$ ./tridentctl upgrade volume default-pvc-2-a8486 -n trident
+----------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
|            NAME      |  SIZE   | STORAGE CLASS | PROTOCOL |
BACKEND UUID             | STATE  | MANAGED |
+----------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
+----------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
```

4. Eseguire un `kubectl describe pv` Per verificare che il volume sia un volume CSI.

```
$ kubectl describe pv default-pvc-2-a8486
Name:            default-pvc-2-a8486
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: csi.trident.netapp.io
                 volume.beta.kubernetes.io/storage-class: standard
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    standard
Status:          Bound
Claim:           default/pvc-2
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        1073741824
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume
source)
    Driver:            csi.trident.netapp.io
    VolumeHandle:      default-pvc-2-a8486
    ReadOnly:          false
    VolumeAttributes:      backendUUID=c5a6f6a4-b052-423b-80d4-
8fb491a14a22

internalName=trid_1907_alpha_default_pvc_2_a8486
                           name=default-pvc-2-a8486
                           protocol=file
Events:                <none>
```

In questo modo, è possibile aggiornare i volumi di tipo NFS/iSCSI creati da Astra Trident al tipo CSI, in base al volume.

Disinstallare Astra Trident

A seconda della modalità di installazione di Astra Trident, esistono diverse opzioni per disinstallarlo.

Disinstallare utilizzando Helm

Se Astra Trident è stato installato utilizzando Helm, è possibile disinstallarlo utilizzando `helm uninstall`.

```
#List the Helm release corresponding to the Astra Trident install.
$ helm ls -n trident
NAME          NAMESPACE       REVISION        UPDATED
STATUS          CHART                           APP VERSION
trident       trident         1               2021-04-20
00:26:42.417764794 +0000 UTC deployed        trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
$ helm uninstall trident -n trident
release "trident" uninstalled
```

Disinstallare utilizzando l'operatore Trident

Se Astra Trident è stato installato utilizzando l'operatore, è possibile disinstallarlo eseguendo una delle seguenti operazioni:

- **Modifica `TridentOrchestrator` Per impostare il flag di disinstallazione:** è possibile modificare `TridentOrchestrator` e impostare `spec.uninstall=true`. Modificare il `TridentOrchestrator` CR e impostare `uninstall` contrassegnare come mostrato di seguito:

  ```
  $  kubectl patch torc <trident-orchestrator-name> --type=merge -p
  '{"spec":{"uninstall":true}}'
  ```

  Quando il `uninstall` flag è impostato su `true`, L'operatore Trident disinstalla Trident, ma non rimuove il TridentOrchestrator stesso. Se si desidera installare di nuovo Trident, è necessario ripulire TridentOrchestrator e crearne uno nuovo.

- **Elimina `TridentOrchestrator`:** rimuovendo il `TridentOrchestrator` CR utilizzato per implementare Astra Trident, si richiede all'operatore di disinstallare Trident. L'operatore elabora la rimozione di `TridentOrchestrator` E procede alla rimozione dell'implementazione e del demonset di Astra Trident, eliminando i pod Trident creati come parte dell'installazione. Per rimuovere completamente Astra Trident (inclusi i CRD creati) e pulire efficacemente l'ardesia pulita, è possibile modificare `TridentOrchestrator` per superare il `wipeout` opzione. Vedere il seguente esempio:

```
$ kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

In questo modo Astra Trident viene disinstallato completamente e vengono cancellati tutti i metadati relativi ai backend e ai volumi gestiti. Le installazioni successive vengono trattate come installazioni nuove.

> Considerare la cancellazione dei CRD solo quando si esegue una disinstallazione completa. Questa operazione non può essere annullata. **Non cancellare i CRD a meno che non si desideri ricominciare e creare una nuova installazione di Astra Trident**.

Disinstallare utilizzando `tridentctl`

Eseguire `uninstall` ingresso comando `tridentctl` Come segue per rimuovere tutte le risorse associate ad Astra Trident, ad eccezione dei CRD e degli oggetti correlati, semplificando l'esecuzione del programma di installazione per l'aggiornamento a una versione più recente.

```
./tridentctl uninstall -n <namespace>
```

Per eseguire una rimozione completa di Astra Trident, rimuovere i finalizzatori dei CRD creati da Astra Trident ed eliminare i CRD.

Downgrade di Astra Trident

Scopri i passaggi necessari per il downgrade a una versione precedente di Astra Trident.

È possibile valutare la possibilità di eseguire il downgrade per diversi motivi, ad esempio:

- Pianificazione di emergenza
- Correzione immediata dei bug osservati in seguito a un aggiornamento
- Problemi di dipendenza, aggiornamenti non riusciti e incompleti

Quando eseguire il downgrade

Quando si esegue il passaggio a una release di Astra Trident che utilizza i CRD, si consiglia di considerare un downgrade. Poiché Astra Trident utilizza ora i CRD per mantenere lo stato, tutte le entità di storage create (backend, classi di storage, PV e snapshot di volumi) hanno oggetti CRD associati invece che dati scritti in `trident` PV (utilizzato dalla versione installata in precedenza di Astra Trident). I PVS, i backend e le classi di storage appena creati vengono mantenuti come oggetti CRD. Se è necessario eseguire il downgrade, questo deve essere tentato solo per una versione di Astra Trident eseguita utilizzando CRD (19.07 e versioni successive). In questo modo si garantisce che tutte le operazioni eseguite sulla release corrente di Astra Trident siano visibili dopo il downgrade.

Quando non eseguire il downgrade

Non eseguire il downgrade a una release di Trident che utilizza `etcd` per mantenere lo stato (19.04 e versioni precedenti). Tutte le operazioni eseguite con la release corrente di Astra Trident non vengono riflesse dopo il downgrade. I PVS appena creati non sono utilizzabili quando si torna a una versione precedente. Le modifiche apportate a oggetti come backend, PVS, classi di storage e snapshot di volumi (create/aggiornate/eliminate) non sono visibili ad Astra Trident quando si torna a una versione precedente. Il ritorno a una versione precedente non interrompe l'accesso ai PVS già creati utilizzando la versione precedente, a meno che non siano stati aggiornati.

Processo di downgrade quando Astra Trident viene installato utilizzando l'operatore

Per le installazioni eseguite con Trident Operator, il processo di downgrade è diverso e non richiede l'utilizzo di `tridentctl`.

Per le installazioni eseguite utilizzando l'operatore Trident, Astra Trident può essere retrocesso a una delle seguenti opzioni:

- Versione installata utilizzando l'operatore namespace-scoped (20.07 - 20.10).
- Versione installata utilizzando l'operatore con ambito cluster (21.01 e versioni successive).

Eseguire il downgrade all'operatore con ambito cluster

Per eseguire il downgrade di Astra Trident a una release che utilizza l'operatore cluster-scoped, attenersi alla procedura indicata di seguito.

Fasi

1. "Disinstallare Astra Trident". **Non estrarre i CRD a meno che non si desideri rimuovere completamente un'installazione esistente.**
2. Eliminare l'operatore con ambito del cluster. A tale scopo, è necessario il manifesto utilizzato per implementare l'operatore. È possibile ottenerlo dal "Trident GitHub repo". Assicurarsi di passare alla filiale desiderata.
3. Continuare a eseguire il downgrade installando la versione desiderata di Astra Trident. Seguire la documentazione relativa alla release desiderata.

Eseguire il downgrade all'operatore con ambito spazio dei nomi

Questa sezione riassume i passaggi necessari per il downgrade a una release Astra Trident compresa tra 20.07 e 20.10, che verrà installata utilizzando l'operatore con ambito namespace.

Fasi

1. "Disinstallare Astra Trident". **Non estrarre i CRD a meno che non si desideri rimuovere completamente un'installazione esistente.** assicurarsi di `tridentorchestrator` viene cancellato.

   ```
   #Check to see if there are any tridentorchestrators present
   $ kubectl get torc
   NAME        AGE
   trident     20h

   #Looks like there is a tridentorchestrator that needs deleting
   $ kubectl delete torc trident
   tridentorchestrator.trident.netapp.io "trident" deleted
   ```

2. Eliminare l'operatore con ambito del cluster. A tale scopo, è necessario il manifesto utilizzato per implementare l'operatore. È possibile ottenerlo qui dal "Trident GitHub repo". Assicurarsi di passare alla filiale desiderata.
3. Eliminare `tridentorchestrator` CRD.

```
#Check to see if ``tridentorchestrators.trident.netapp.io`` CRD is
present and delete it.
$ kubectl get crd tridentorchestrators.trident.netapp.io
NAME                                        CREATED AT
tridentorchestrators.trident.netapp.io   2021-01-21T21:11:37Z
$ kubectl delete crd tridentorchestrators.trident.netapp.io
customresourcedefinition.apiextensions.k8s.io
"tridentorchestrators.trident.netapp.io" deleted
```

Astra Trident è stato disinstallato.

4. Continuare con il downgrade installando la versione desiderata. Seguire la documentazione relativa alla release desiderata.

Eseguire il downgrade utilizzando Helm

Per eseguire il downgrade, utilizzare `helm rollback` comando. Vedere il seguente esempio:

```
$ helm rollback trident [revision #]
```

Processo di downgrade quando Astra Trident viene installato mediante `tridentctl`

Se Astra Trident è stato installato utilizzando `tridentctl`, il processo di downgrade prevede i seguenti passaggi. Questa sequenza illustra il processo di downgrade per passare da Astra Trident 21.07 a 20.07.

> Prima di iniziare il downgrade, è necessario creare un'istantanea del cluster Kubernetes `etcd`. Ciò consente di eseguire il backup dello stato corrente dei CRD di Astra Trident.

Fasi

1. Assicurarsi che Trident sia installato utilizzando `tridentctl`. Se non si è sicuri di come sia installato Astra Trident, eseguire questo semplice test:
   a. Elencare i pod presenti nello spazio dei nomi Trident.
   b. Identificare la versione di Astra Trident in esecuzione nel cluster. È possibile utilizzare entrambi `tridentctl` Oppure guarda l'immagine utilizzata nei pod Trident.
   c. Se **non viene visualizzato** A. `tridentOrchestrator`, (o) a. `tridentprovisioner`, (o) un pod denominato `trident-operator-xxxxxxxxxx-xxxxx`, Astra Trident **è installato** con `tridentctl`.
2. Disinstallare Astra Trident con l'esistente `tridentctl` binario. In questo caso, verrà disinstallato con il file binario 21.07.

```
$ tridentctl version -n trident
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.0        | 21.07.0        |
+----------------+----------------+

$ tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted Trident daemonset.
INFO Deleted Trident service.
INFO Deleted Trident secret.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Deleted pod security policy.
podSecurityPolicy=tridentpods
INFO The uninstaller did not delete Trident's namespace in case it is
going to be reused.
INFO Trident uninstallation succeeded.
```

3. Una volta completata questa operazione, ottenere il binario Trident per la versione desiderata (in questo esempio, 20.07) e utilizzarlo per installare Astra Trident. È possibile generare YAML personalizzati per a. "installazione personalizzata" se necessario.

```
$ cd 20.07/trident-installer/
$ ./tridentctl install -n trident-ns
INFO Created installer service account.
serviceaccount=trident-installer
INFO Created installer cluster role.               clusterrole=trident-
installer
INFO Created installer cluster role binding.
clusterrolebinding=trident-installer
INFO Created installer configmap.                  configmap=trident-
installer
...
...
INFO Deleted installer cluster role binding.
INFO Deleted installer cluster role.
INFO Deleted installer service account.
```

Il processo di downgrade è completo.

Utilizzare Astra Trident

Configurare i backend

Un backend definisce la relazione tra Astra Trident e un sistema storage. Spiega ad Astra Trident come comunicare con quel sistema storage e come Astra Trident dovrebbe eseguire il provisioning dei volumi da esso. Astra Trident offrirà automaticamente pool di storage da backend che insieme soddisfano i requisiti definiti da una classe di storage. Scopri di più sulla configurazione del back-end in base al tipo di sistema storage in uso.

- "Configurare un backend Azure NetApp Files"
- "Configurare un Cloud Volumes Service per il backend della piattaforma cloud Google"
- "Configurare un backend NetApp HCI o SolidFire"
- "Configurare un backend con driver NAS ONTAP o Cloud Volumes ONTAP"
- "Configurare un backend con i driver SAN ONTAP o Cloud Volumes ONTAP"
- "Utilizza Astra Trident con Amazon FSX per NetApp ONTAP"

Configurare un backend Azure NetApp Files

Scopri come configurare Azure NetApp Files (ANF) come backend per l'installazione di Astra Trident utilizzando le configurazioni di esempio fornite.

> Il servizio Azure NetApp Files non supporta volumi inferiori a 100 GB. Astra Trident crea automaticamente volumi da 100 GB se viene richiesto un volume più piccolo.

Di cosa hai bisogno

Per configurare e utilizzare un "Azure NetApp Files" back-end, sono necessari i seguenti elementi:

- `subscriptionID` Da un abbonamento Azure con Azure NetApp Files attivato.
- `tenantID`, `clientID`, e. `clientSecret` da un "Registrazione dell'app" In Azure Active Directory con autorizzazioni sufficienti per il servizio Azure NetApp Files. La registrazione dell'applicazione deve utilizzare `Owner` oppure `Contributor` Ruolo predefinito da Azure.

  > Per ulteriori informazioni sui ruoli integrati di Azure, consulta la "Documentazione di Azure".

- Azure `location` che ne contiene almeno uno "subnet delegata". A partire da Trident 22.01, il `location` parametro è un campo obbligatorio al livello superiore del file di configurazione back-end. I valori di posizione specificati nei pool virtuali vengono ignorati.
- Se si utilizza Azure NetApp Files per la prima volta o in una nuova posizione, è necessaria una configurazione iniziale. Vedere "guida rapida".

A proposito di questa attività

In base alla configurazione del backend (subnet, rete virtuale, livello di servizio e posizione), Trident crea volumi ANF su pool di capacità disponibili nella posizione richiesta e corrispondenti al livello di servizio e alla subnet richiesti.

> NOTA: Astra Trident non supporta i pool di capacità QoS manuali.

Opzioni di configurazione back-end

Per le opzioni di configurazione del backend, consultare la tabella seguente:

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di storage	"azure-netapp-files"
`backendName`	Nome personalizzato o backend dello storage	Nome del driver + "_" + caratteri casuali
`subscriptionID`	L'ID dell'abbonamento dell'abbonamento Azure	
`tenantID`	L'ID tenant di una registrazione app	
`clientID`	L'ID client di una registrazione dell'applicazione	
`clientSecret`	Il segreto del client da una registrazione dell'applicazione	
`serviceLevel`	Uno di `Standard`, `Premium`, o. `Ultra`	"" (casuale)
`location`	Nome della posizione di Azure in cui verranno creati i nuovi volumi	
`serviceLevel`	Uno di `Standard`, `Premium`, o. `Ultra`	"" (casuale)
`resourceGroups`	Elenco dei gruppi di risorse per filtrare le risorse rilevate	"[]" (nessun filtro)
`netappAccounts`	Elenco degli account NetApp per il filtraggio delle risorse rilevate	"[]" (nessun filtro)
`capacityPools`	Elenco dei pool di capacità per filtrare le risorse rilevate	"[]" (nessun filtro, casuale)
`virtualNetwork`	Nome di una rete virtuale con una subnet delegata	""
`subnet`	Nome di una subnet delegata a. `Microsoft.Netapp/volumes`	""
`nfsMountOptions`	Controllo dettagliato delle opzioni di montaggio NFS.	"nfsvers=3"
`limitVolumeSize`	Il provisioning non riesce se le dimensioni del volume richiesto sono superiori a questo valore	"" (non applicato per impostazione predefinita)

Parametro	Descrizione	Predefinito
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Esempio, `\{"api": false, "method": true, "discovery": true}`. Non utilizzare questa opzione a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.	nullo

Se viene visualizzato l'errore "Nessun pool di capacità trovato" quando si tenta di creare un PVC, è probabile che la registrazione dell'applicazione non abbia le autorizzazioni e le risorse richieste (subnet, rete virtuale, pool di capacità) associate. Astra Trident registrerà le risorse Azure rilevate al momento della creazione del backend quando il debug è attivato. Assicurarsi di controllare se viene utilizzato un ruolo appropriato.

Se si desidera montare i volumi utilizzando NFS versione 4.1, è possibile includere `nfsvers=4` Nell'elenco delle opzioni di montaggio delimitate da virgole, scegliere NFS v4.1. Tutte le opzioni di montaggio impostate in una classe di storage sovrascrivono le opzioni di montaggio impostate in un file di configurazione back-end.

I valori per `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork`, e. `subnet` può essere specificato utilizzando nomi brevi o pienamente qualificati. I nomi brevi possono corrispondere a più risorse con lo stesso nome, pertanto si consiglia di utilizzare nomi completi nella maggior parte delle situazioni. Il `resourceGroups`, `netappAccounts`, e. `capacityPools` i valori sono filtri che limitano l'insieme di risorse rilevate a quelle disponibili per questo backend di storage e possono essere specificati in qualsiasi combinazione. I nomi pienamente qualificati sono del seguente formato:

Tipo	Formato
Gruppo di risorse	<resource group>
Account NetApp	<resource group>/<netapp account>
Pool di capacità	<resource group>/<netapp account>/<capacity pool>
Rete virtuale	<resource group>/<virtual network>
Subnet	<resource group>/<virtual network>/<subnet>

Per impostazione predefinita, è possibile controllare il provisioning di ciascun volume specificando le seguenti opzioni in una sezione speciale del file di configurazione. Vedere gli esempi di configurazione riportati di seguito.

Parametro	Descrizione	Predefinito
`exportRule`	Regola o regole di esportazione per i nuovi volumi	"0.0.0.0/0"
`snapshotDir`	Controlla la visibilità della directory .snapshot	"falso"

Parametro	Descrizione	Predefinito
`size`	La dimensione predefinita dei nuovi volumi	"100 G"
`unixPermissions`	Le autorizzazioni unix dei nuovi volumi (4 ottali cifre)	"" (funzione di anteprima, richiede la whitelist nell'abbonamento)

Il `exportRule` Il valore deve essere un elenco separato da virgole di qualsiasi combinazione di indirizzi IPv4 o subnet IPv4 nella notazione CIDR.

> Per tutti i volumi creati su un backend ANF, Astra Trident copia tutte le etichette presenti su un pool di storage nel volume di storage al momento del provisioning. Gli amministratori dello storage possono definire le etichette per ogni pool di storage e raggruppare tutti i volumi creati in un pool di storage. In questo modo è possibile differenziare i volumi in base a una serie di etichette personalizzabili fornite nella configurazione di back-end.

Esempio 1: Configurazione minima

Questa è la configurazione backend minima assoluta. Con questa configurazione, Astra Trident rileva tutti gli account NetApp, i pool di capacità e le subnet delegate ad ANF nella posizione configurata e inserisce i nuovi volumi in uno di questi pool e sottoreti in modo casuale.

Questa configurazione è ideale quando si inizia a utilizzare ANF e si provano le cose, ma in pratica si desidera fornire un ambito aggiuntivo per i volumi che si esegue il provisioning.

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
    "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
    "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
    "clientSecret": "SECRET",
    "location": "eastus"
}
```

Esempio 2: Configurazione specifica del livello di servizio con filtri del pool di capacità

Questa configurazione di back-end consente di posizionare i volumi in Azure `eastus` posizione in un `Ultra` pool di capacità. Astra Trident rileva automaticamente tutte le subnet delegate ad ANF in quella posizione e inserisce un nuovo volume su una di esse in modo casuale.

```
    {
        "version": 1,
        "storageDriverName": "azure-netapp-files",
        "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
        "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
        "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
        "clientSecret": "SECRET",
        "location": "eastus",
        "serviceLevel": "Ultra",
        "capacityPools": [
            "application-group-1/account-1/ultra-1",
            "application-group-1/account-1/ultra-2"
],
    }
```

Esempio 3: Configurazione avanzata

Questa configurazione di back-end riduce ulteriormente l'ambito del posizionamento del volume in una singola subnet e modifica alcune impostazioni predefinite di provisioning del volume.

```
    {
        "version": 1,
        "storageDriverName": "azure-netapp-files",
        "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
        "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
        "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
        "clientSecret": "SECRET",
        "location": "eastus",
        "serviceLevel": "Ultra",
        "capacityPools": [
            "application-group-1/account-1/ultra-1",
            "application-group-1/account-1/ultra-2"
],
        "virtualNetwork": "my-virtual-network",
        "subnet": "my-subnet",
        "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
        "limitVolumeSize": "500Gi",
        "defaults": {
            "exportRule": "10.0.0.0/24,10.0.1.0/24,10.0.2.100",
            "snapshotDir": "true",
            "size": "200Gi",
            "unixPermissions": "0777"
=======
        }
    }
```

Esempio 4: Configurazione del pool di storage virtuale

Questa configurazione di back-end definisce più pool di storage in un singolo file. Ciò è utile quando si dispone di più pool di capacità che supportano diversi livelli di servizio e si desidera creare classi di storage in Kubernetes che ne rappresentano.

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
    "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
    "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
    "clientSecret": "SECRET",
    "location": "eastus",
    "resourceGroups": ["application-group-1"],
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
    "labels": {
        "cloud": "azure"
    },
    "location": "eastus",

    "storage": [
        {
            "labels": {
                "performance": "gold"
            },
            "serviceLevel": "Ultra",
            "capacityPools": ["ultra-1", "ultra-2"]
        },
        {
            "labels": {
                "performance": "silver"
            },
            "serviceLevel": "Premium",
            "capacityPools": ["premium-1"]
        },
        {
            "labels": {
                "performance": "bronze"
            },
            "serviceLevel": "Standard",
            "capacityPools": ["standard-1", "standard-2"]
        }
    ]
}
```

Quanto segue `StorageClass` le definizioni si riferiscono ai pool di storage sopra indicati. Utilizzando `parameters.selector` è possibile specificare per ciascun campo `StorageClass` il pool di visualizzazioni utilizzato per ospitare un volume. Gli aspetti del volume saranno definiti nel pool selezionato.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze"
allowVolumeExpansion: true
```

Quali sono le prossime novità?

Dopo aver creato il file di configurazione back-end, eseguire il seguente comando:

```
tridentctl create backend -f <backend-file>
```

Se la creazione del backend non riesce, si è verificato un errore nella configurazione del backend. È possibile visualizzare i log per determinare la causa eseguendo il seguente comando:

```
tridentctl logs
```

Dopo aver identificato e corretto il problema con il file di configurazione, è possibile eseguire nuovamente il comando create.

Configurare un CVS per il backend GCP

Scopri come configurare NetApp Cloud Volumes Service (CVS) per la piattaforma cloud Google (GCP) come

back-end per l'installazione di Astra Trident utilizzando le configurazioni di esempio fornite.

> NetApp Cloud Volumes Service per Google Cloud non supporta volumi CVS-Performance di dimensioni inferiori a 100 GiB o volumi CVS di dimensioni inferiori a 300 GiB. Astra Trident crea automaticamente volumi di dimensioni minime se il volume richiesto è inferiore alla dimensione minima.

Di cosa hai bisogno

Per configurare e utilizzare "Cloud Volumes Service per Google Cloud" back-end, sono necessari i seguenti elementi:

- Un account Google Cloud configurato con NetApp CVS
- Numero di progetto dell'account Google Cloud
- Account di servizio Google Cloud con `netappcloudvolumes.admin` ruolo
- File delle chiavi API per l'account del servizio CVS

Astra Trident ora include il supporto per volumi più piccoli con l'impostazione predefinita "Tipo di servizio CVS su GCP". Per i backend creati con `storageClass=software`, I volumi avranno ora una dimensione di provisioning minima di 300 GiB. CVS attualmente fornisce questa funzione in disponibilità controllata e non fornisce supporto tecnico. Gli utenti devono iscriversi per accedere ai volumi inferiori a 1 TiB "qui". NetApp consiglia ai clienti di consumare volumi inferiori a 1 TiB per carichi di lavoro **non in produzione**.

> Quando si implementano backend utilizzando il tipo di servizio CVS predefinito (`storageClass=software`), gli utenti devono ottenere l'accesso alla funzione volumi sub-1TiB su GCP per i numeri di progetto e gli ID progetto in questione. Questo è necessario per Astra Trident per eseguire il provisioning di volumi inferiori a 1 TiB. In caso contrario, le creazioni di volumi non verranno superate per PVC inferiori a 600 GiB. Ottenere l'accesso a volumi inferiori a 1 TiB utilizzando "questo modulo".

I volumi creati da Astra Trident per il livello di servizio CVS predefinito verranno forniti come segue:

- I PVC di dimensioni inferiori a 300 GiB determinano la creazione di un volume CVS da 300 GiB da parte di Astra Trident.
- I PVC compresi tra 300 GiB e 600 GiB determineranno la creazione di un volume CVS della dimensione richiesta da parte di Astra Trident.
- I PVC compresi tra 600 GiB e 1 TIB determineranno la creazione di un volume CVS 1TiB da parte di Astra Trident.
- I PVC superiori a 1 TIB determineranno la creazione da parte di Astra Trident di un volume CVS della dimensione richiesta.

Opzioni di configurazione back-end

Per le opzioni di configurazione del backend, consultare la tabella seguente:

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di storage	"gcp-cvs"

Parametro	Descrizione	Predefinito
`backendName`	Nome personalizzato o backend dello storage	Nome del driver + "_" + parte della chiave API
`storageClass`	Tipo di storage. Scegliere tra `hardware` (prestazioni ottimizzate) o. `software` (Tipo di servizio CVS)	
`projectNumber`	Numero di progetto dell'account Google Cloud. Il valore si trova nella home page del portale Google Cloud.	
`apiRegion`	Regione dell'account CVS. È la regione in cui il back-end eseguirà il provisioning dei volumi.	
`apiKey`	Chiave API per l'account del servizio Google Cloud con `netappcloudvolumes.admin` ruolo. Include il contenuto in formato JSON di un file di chiave privata dell'account di un servizio Google Cloud (copia integrale nel file di configurazione del backend).	
`proxyURL`	URL del proxy se il server proxy ha richiesto di connettersi all'account CVS. Il server proxy può essere un proxy HTTP o un proxy HTTPS. Per un proxy HTTPS, la convalida del certificato viene ignorata per consentire l'utilizzo di certificati autofirmati nel server proxy. I server proxy con autenticazione abilitata non sono supportati.	
`nfsMountOptions`	Controllo dettagliato delle opzioni di montaggio NFS.	"nfsvers=3"
`limitVolumeSize`	Il provisioning non riesce se le dimensioni del volume richiesto sono superiori a questo valore	"" (non applicato per impostazione predefinita)
`serviceLevel`	Il livello di servizio CVS per i nuovi volumi. I valori sono "standard", "premium" e "Extreme".	"standard"
`network`	Rete GCP utilizzata per volumi CVS	"predefinito"

Parametro	Descrizione	Predefinito
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Esempio, `\{"api":false, "method":true}`. Non utilizzare questa opzione a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.	nullo

Se si utilizza una rete VPC condivisa, entrambi `projectNumber` e. `hostProjectNumber` deve essere specificato. In tal caso, `projectNumber` è il progetto di servizio, e. `hostProjectNumber` è il progetto host.

Il `apiRegion` Rappresenta la regione GCP in cui Astra Trident crea volumi CVS. Durante la creazione di cluster Kubernetes a più aree, i volumi CVS vengono creati in un `apiRegion` Può essere utilizzato nei carichi di lavoro pianificati su nodi in più regioni GCP. Tenere presente che il traffico interregionale comporta un costo aggiuntivo.

- Per abilitare l'accesso multi-regione, la definizione StorageClass per `allowedTopologies` deve includere tutte le regioni. Ad esempio:

```
- key: topology.kubernetes.io/region
  values:
  - us-east1
  - europe-west1
```

- `storageClass` è un parametro facoltativo che è possibile utilizzare per selezionare il desiderato "Tipo di servizio CVS". È possibile scegliere tra il tipo di servizio CVS di base (`storageClass=software`) O il tipo di servizio CVS-Performance (`storageClass=hardware`), che Trident utilizza per impostazione predefinita. Assicurarsi di specificare un `apiRegion` Che fornisce il rispettivo CVS `storageClass` nella definizione di back-end.

L'integrazione di Astra Trident con il tipo di servizio CVS di base su Google Cloud è una funzionalità **beta**, non destinata ai carichi di lavoro di produzione. Trident è **completamente supportato** con il tipo di servizio CVS-Performance e lo utilizza per impostazione predefinita.

Ogni back-end esegue il provisioning dei volumi in una singola area di Google Cloud. Per creare volumi in altre regioni, è possibile definire backend aggiuntivi.

Per impostazione predefinita, è possibile controllare il provisioning di ciascun volume specificando le seguenti opzioni in una sezione speciale del file di configurazione. Vedere gli esempi di configurazione riportati di seguito.

Parametro	Descrizione	Predefinito
`exportRule`	Regola o regole di esportazione per i nuovi volumi	"0.0.0.0/0"
`snapshotDir`	Accesso a `.snapshot` directory	"falso"

Parametro	Descrizione	Predefinito
`snapshotReserve`	Percentuale di volume riservato agli snapshot	"" (accettare CVS come valore predefinito 0)
`size`	Le dimensioni dei nuovi volumi	"100 Gi"

Il `exportRule` Il valore deve essere un elenco separato da virgole di qualsiasi combinazione di indirizzi IPv4 o subnet IPv4 nella notazione CIDR.

> Per tutti i volumi creati su un backend CVS Google Cloud, Trident copia tutte le etichette presenti su un pool di storage nel volume di storage al momento del provisioning. Gli amministratori dello storage possono definire le etichette per ogni pool di storage e raggruppare tutti i volumi creati in un pool di storage. In questo modo è possibile differenziare i volumi in base a una serie di etichette personalizzabili fornite nella configurazione di back-end.

Esempio 1: Configurazione minima

Questa è la configurazione backend minima assoluta.

```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "apiRegion": "us-west2",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "1234567890123456789012345678901234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisI
sAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSa
PIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZN
chRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlz
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHi
sIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOgu
SaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyA
ZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz
lzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrt
```

```
HisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbO
guSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKe
yAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRA
GzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    }
}
```

Esempio 2: Configurazione del tipo di servizio CVS di base

Questo esempio mostra una definizione di back-end che utilizza il tipo di servizio CVS di base, che è destinato ai carichi di lavoro generici e fornisce performance leggere/moderate, insieme ad un'elevata disponibilità zonale.

```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "storageClass": "software",
    "apiRegion": "us-east4",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "1234567890123456789012345678901234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisI
sAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSa
PIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZN
chRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlz
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHi
```

```
sIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOgu
SaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyA
ZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz
lzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrt
HisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbO
guSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKe
yAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRA
GzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    }
}
```


Esempio 3: Configurazione a livello di servizio singolo

Questo esempio mostra un file backend che applica gli stessi aspetti a tutti gli storage creati da Astra Trident nella regione di Google Cloud us-west2. Questo esempio mostra anche l'utilizzo di `proxyURL` nel file di configurazione back-end.


```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "apiRegion": "us-west2",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "1234567890123456789012345678901234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisI
sAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSa
```

```
PIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZN
chRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlz
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHi
sIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOgu
SaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyA
ZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz
lzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrt
HisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbO
guSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKe
yAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRA
GzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    },
    "proxyURL": "http://proxy-server-hostname/",
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
    "limitVolumeSize": "10Ti",
    "serviceLevel": "premium",
    "defaults": {
        "snapshotDir": "true",
        "snapshotReserve": "5",
        "exportRule": "10.0.0.0/24,10.0.1.0/24,10.0.2.100",
        "size": "5Ti"
    }
}
```

Esempio 4: Configurazione del pool di storage virtuale

Questo esempio mostra il file di definizione back-end configurato con i pool di storage virtuali insieme a. `StorageClasses` che fanno riferimento a loro.

Nel file di definizione del backend di esempio mostrato di seguito, vengono impostati valori predefiniti specifici per tutti i pool di storage, che impostano `snapshotReserve` al 5% e a. `exportRule` a 0.0.0.0/0. I pool di storage virtuali sono definiti in `storage` sezione. In questo esempio, ogni singolo pool di storage imposta il proprio `serviceLevel`e alcuni pool sovrascrivono i valori predefiniti.

```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "apiRegion": "us-west2",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "1234567890123456789012345678901234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisI
sAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSa
PIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZN
chRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlz
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHi
sIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOgu
SaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyA
ZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz
lzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrt
HisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbO
guSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKe
yAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRA
GzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
```

```
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    },
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",

    "defaults": {
        "snapshotReserve": "5",
        "exportRule": "0.0.0.0/0"
    },

    "labels": {
        "cloud": "gcp"
    },
    "region": "us-west2",

    "storage": [
        {
            "labels": {
                "performance": "extreme",
                "protection": "extra"
            },
            "serviceLevel": "extreme",
            "defaults": {
                "snapshotDir": "true",
                "snapshotReserve": "10",
                "exportRule": "10.0.0.0/24"
            }
        },
        {
            "labels": {
                "performance": "extreme",
                "protection": "standard"
            },
            "serviceLevel": "extreme"
        },
        {
            "labels": {
                "performance": "premium",
                "protection": "extra"
            },
            "serviceLevel": "premium",
```

```
            "defaults": {
                "snapshotDir": "true",
                "snapshotReserve": "10"
            }
        },

        {
            "labels": {
                "performance": "premium",
                "protection": "standard"
            },
            "serviceLevel": "premium"
        },

        {
            "labels": {
                "performance": "standard"
            },
            "serviceLevel": "standard"
        }
    ]
}
```

Le seguenti definizioni di StorageClass si riferiscono ai pool di storage di cui sopra. Utilizzando `parameters.selector` È possibile specificare per ogni StorageClass il pool virtuale utilizzato per ospitare un volume. Gli aspetti del volume saranno definiti nel pool selezionato.

Il primo StorageClass (`cvs-extreme-extra-protection`) viene mappato al primo pool di storage virtuale. Questo è l'unico pool che offre performance estreme con una riserva di snapshot del 10%. L'ultima StorageClass (`cvs-extra-protection`) richiama qualsiasi pool di storage che fornisce una riserva di snapshot del 10%. Astra Trident decide quale pool di storage virtuale è selezionato e garantisce che il requisito di riserva snapshot sia soddisfatto.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=extreme; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: netapp.io/trident
```

```
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
provisioner: netapp.io/trident
parameters:
  selector: "performance=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "protection=extra"
allowVolumeExpansion: true
```

Quali sono le prossime novità?

Dopo aver creato il file di configurazione back-end, eseguire il seguente comando:

```
tridentctl create backend -f <backend-file>
```

Se la creazione del backend non riesce, si è verificato un errore nella configurazione del backend. È possibile visualizzare i log per determinare la causa eseguendo il seguente comando:

```
tridentctl logs
```

Dopo aver identificato e corretto il problema con il file di configurazione, è possibile eseguire nuovamente il comando create.

Configurare un backend NetApp HCI o SolidFire

Scopri come creare e utilizzare un backend Element con l'installazione di Astra Trident.

Di cosa hai bisogno

- Un sistema storage supportato che esegue il software Element.
- Credenziali per un amministratore del cluster NetApp HCI/SolidFire o un utente tenant in grado di gestire i volumi.
- Tutti i nodi di lavoro di Kubernetes devono disporre dei tool iSCSI appropriati. Vedere "informazioni sulla preparazione del nodo di lavoro".

Cosa devi sapere

Il `solidfire-san` il driver di storage supporta entrambe le modalità di volume: file e block. Per `Filesystem` VolumeMode, Astra Trident crea un volume e un filesystem. Il tipo di file system viene specificato da StorageClass.

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
`solidfire-san`	ISCSI	Blocco	RWO, ROX, RWX	Nessun filesystem. Dispositivo a blocchi raw.
`solidfire-san`	ISCSI	Blocco	RWO, ROX, RWX	Nessun filesystem. Dispositivo a blocchi raw.
`solidfire-san`	ISCSI	Filesystem	RWO, ROX	`xfs`, `ext3`, `ext4`
`solidfire-san`	ISCSI	Filesystem	RWO, ROX	`xfs`, `ext3`, `ext4`

Astra Trident utilizza CHAP quando funziona come provider CSI avanzato. Se si utilizza CHAP (che è l'impostazione predefinita per CSI), non sono necessarie ulteriori operazioni di preparazione. Si consiglia di impostare in modo esplicito `UseCHAP` Opzione per utilizzare CHAP con Trident non CSI. In caso contrario, vedere "qui".

I gruppi di accesso ai volumi sono supportati solo dal framework convenzionale non CSI per Astra Trident. Se configurato per funzionare in modalità CSI, Astra Trident utilizza CHAP.

In caso contrario `AccessGroups` oppure `UseCHAP` viene impostata una delle seguenti regole:

- Se l'impostazione predefinita `trident` viene rilevato un gruppo di accesso, vengono utilizzati i gruppi di accesso.
- Se non viene rilevato alcun gruppo di accesso e Kubernetes versione 1.7 o successiva, viene utilizzato CHAP.

Opzioni di configurazione back-end

Per le opzioni di configurazione del backend, consultare la tabella seguente:

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di storage	"Solidfire-san"
`backendName`	Nome personalizzato o backend dello storage	"SolidFire_" + indirizzo IP dello storage (iSCSI)
`Endpoint`	MVIP per il cluster SolidFire con credenziali tenant	
`SVIP`	Porta e indirizzo IP dello storage (iSCSI)	
`labels`	Set di etichette arbitrarie formattate con JSON da applicare sui volumi.	""
`TenantName`	Nome tenant da utilizzare (creato se non trovato)	
`InitiatorIFace`	Limitare il traffico iSCSI a un'interfaccia host specifica	"predefinito"
`UseCHAP`	Utilizzare CHAP per autenticare iSCSI	vero
`AccessGroups`	Elenco degli ID del gruppo di accesso da utilizzare	Trova l'ID di un gruppo di accesso denominato "tridente"
`Types`	Specifiche QoS	
`limitVolumeSize`	Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore	"" (non applicato per impostazione predefinita)
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Ad esempio, {"api":false,} method":true	nullo

Non utilizzare `debugTraceFlags` a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.

Per tutti i volumi creati, Astra Trident copia tutte le etichette presenti in un pool di storage nel LUN dello storage di backup al momento del provisioning. Gli amministratori dello storage possono definire le etichette per ogni pool di storage e raggruppare tutti i volumi creati in un pool di storage. In questo modo è possibile differenziare i volumi in base a una serie di etichette personalizzabili fornite nella configurazione di back-end.

Esempio 1: Configurazione back-end per `solidfire-san` driver con tre tipi di volume

Questo esempio mostra un file backend che utilizza l'autenticazione CHAP e modellazione di tre tipi di volume con specifiche garanzie di QoS. È molto probabile che si definiscano le classi di storage per utilizzarle utilizzando `IOPS` parametro della classe di storage.

```
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://<user>:<password>@<mvip>/json-rpc/8.0",
    "SVIP": "<svip>:3260",
    "TenantName": "<tenant>",
    "labels": {"k8scluster": "dev1", "backend": "dev1-element-cluster"},
    "UseCHAP": true,
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS": 2000,
"burstIOPS": 4000}},
              {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS": 6000,
"burstIOPS": 8000}},
              {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS": 8000,
"burstIOPS": 10000}}]
}
```

Esempio 2: Configurazione del backend e della classe di storage per `solidfire-san` driver con pool di storage virtuali

Questo esempio mostra il file di definizione back-end configurato con i pool di storage virtuali insieme a StorageClasses che fanno riferimento a questi.

Nel file di definizione del backend di esempio mostrato di seguito, vengono impostati valori predefiniti specifici per tutti i pool di storage, che impostano `type` In Silver. I pool di storage virtuali sono definiti in `storage` sezione. In questo esempio, alcuni pool di storage impostano il proprio tipo e alcuni pool sovrascrivono i valori predefiniti precedentemente impostati.

```
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://<user>:<password>@<mvip>/json-rpc/8.0",
    "SVIP": "<svip>:3260",
    "TenantName": "<tenant>",
    "UseCHAP": true,
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS": 2000,
"burstIOPS": 4000}},
              {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS": 6000,
"burstIOPS": 8000}},
              {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS": 8000,
"burstIOPS": 10000}}],

    "type": "Silver",
    "labels":{"store":"solidfire", "k8scluster": "dev-1-cluster"},
    "region": "us-east-1",

    "storage": [
        {
            "labels":{"performance":"gold", "cost":"4"},
            "zone":"us-east-1a",
            "type":"Gold"
        },
        {
            "labels":{"performance":"silver", "cost":"3"},
            "zone":"us-east-1b",
            "type":"Silver"
        },
        {
            "labels":{"performance":"bronze", "cost":"2"},
            "zone":"us-east-1c",
            "type":"Bronze"
        },
        {
            "labels":{"performance":"silver", "cost":"1"},
            "zone":"us-east-1d"
        }
    ]
}
```

Le seguenti definizioni di StorageClass si riferiscono ai pool di storage virtuali sopra indicati. Utilizzando il `parameters.selector` Ciascun StorageClass richiama i pool virtuali che possono essere utilizzati per ospitare un volume. Gli aspetti del volume saranno definiti nel pool virtuale scelto.

Il primo StorageClass (`solidfire-gold-four`) verrà mappato al primo pool di storage virtuale. Questo è

l'unico pool che offre performance eccellenti con un `Volume Type QoS` Dell'oro. L'ultima StorageClass (`solidfire-silver`) definisce qualsiasi pool di storage che offra performance di livello silver. Astra Trident deciderà quale pool di storage virtuale è selezionato e garantirà il rispetto dei requisiti di storage.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold; cost=4"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=3"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze; cost=2"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=1"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
  fsType: "ext4"
```

Trova ulteriori informazioni

- "Gruppi di accesso ai volumi"

Configurare un backend con i driver SAN ONTAP o Cloud Volumes ONTAP

Informazioni sulla configurazione di un backend ONTAP con driver SAN ONTAP e Cloud Volumes ONTAP.

- "Preparazione"
- "Configurazione ed esempi"

Autorizzazioni utente

Astra Trident prevede di essere eseguito come amministratore di ONTAP o SVM, in genere utilizzando `admin` utente del cluster o un `vsadmin` Utente SVM o un utente con un nome diverso che ha lo stesso ruolo. Per le implementazioni di Amazon FSX per NetApp ONTAP, Astra Trident prevede di essere eseguito come amministratore di ONTAP o SVM, utilizzando il cluster `fsxadmin` utente o a. `vsadmin` Utente SVM o un utente con un nome diverso che ha lo stesso ruolo. Il `fsxadmin` user è un sostituto limitato per l'utente amministratore del cluster.

> Se si utilizza `limitAggregateUsage` parametro, sono richieste le autorizzazioni di amministrazione del cluster. Quando si utilizza Amazon FSX per NetApp ONTAP con Astra Trident, il `limitAggregateUsage` il parametro non funziona con `vsadmin` e. `fsxadmin` account utente. L'operazione di configurazione non riesce se si specifica questo parametro.

Sebbene sia possibile creare un ruolo più restrittivo all'interno di ONTAP che un driver Trident può utilizzare, non lo consigliamo. La maggior parte delle nuove release di Trident chiamerà API aggiuntive che dovrebbero essere considerate, rendendo gli aggiornamenti difficili e soggetti a errori.

Preparazione

Scopri come preparare la configurazione di un backend ONTAP con i driver SAN ONTAP. Per tutti i backend ONTAP, Astra Trident richiede almeno un aggregato assegnato alla SVM.

È inoltre possibile eseguire più di un driver e creare classi di storage che puntino all'una o all'altra. Ad esempio, è possibile configurare un `san-dev` classe che utilizza `ontap-san` driver e a. `san-default` classe che utilizza `ontap-san-economy` uno.

Tutti i nodi di lavoro di Kubernetes devono disporre dei tool iSCSI appropriati. Vedere "qui" per ulteriori dettagli.

Autenticazione

Astra Trident offre due modalità di autenticazione di un backend ONTAP.

- Basato sulle credenziali: Nome utente e password di un utente ONTAP con le autorizzazioni richieste. Si consiglia di utilizzare un ruolo di accesso di sicurezza predefinito, ad esempio `admin` oppure `vsadmin` Per garantire la massima compatibilità con le versioni di ONTAP.
- Basato su certificato: Astra Trident può anche comunicare con un cluster ONTAP utilizzando un certificato installato sul backend. In questo caso, la definizione di backend deve contenere i valori codificati in Base64 del certificato client, della chiave e del certificato CA attendibile, se utilizzato (consigliato).

Gli utenti possono anche scegliere di aggiornare i back-end esistenti, scegliendo di passare da basato su credenziali a basato su certificato e viceversa. Se **vengono forniti sia credenziali che certificati**, Astra

Trident utilizza per impostazione predefinita i certificati mentre emette un avviso per rimuovere le credenziali dalla definizione di backend.

Abilitare l'autenticazione basata su credenziali

Astra Trident richiede le credenziali di un amministratore con ambito SVM/cluster per comunicare con il backend ONTAP. Si consiglia di utilizzare ruoli standard predefiniti, ad esempio `admin` oppure `vsadmin`. Ciò garantisce la compatibilità con le future release di ONTAP che potrebbero esporre le API delle funzionalità da utilizzare nelle future release di Astra Trident. È possibile creare e utilizzare un ruolo di accesso di sicurezza personalizzato con Astra Trident, ma non è consigliato.

Una definizione di back-end di esempio avrà un aspetto simile al seguente:

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "secret",
}
```

Tenere presente che la definizione di backend è l'unica posizione in cui le credenziali vengono memorizzate in testo normale. Una volta creato il backend, i nomi utente e le password vengono codificati con Base64 e memorizzati come segreti Kubernetes. La creazione/l'updation di un backend è l'unico passaggio che richiede la conoscenza delle credenziali. Pertanto, si tratta di un'operazione di sola amministrazione, che deve essere eseguita dall'amministratore Kubernetes/storage.

Abilitare l'autenticazione basata su certificato

I backend nuovi ed esistenti possono utilizzare un certificato e comunicare con il backend ONTAP. Nella definizione di backend sono necessari tre parametri.

- ClientCertificate: Valore del certificato client codificato con base64.
- ClientPrivateKey: Valore codificato in base64 della chiave privata associata.
- TrustedCACertificate: Valore codificato in base64 del certificato CA attendibile. Se si utilizza una CA attendibile, è necessario fornire questo parametro. Questa operazione può essere ignorata se non viene utilizzata alcuna CA attendibile.

Un workflow tipico prevede i seguenti passaggi.

Fasi

1. Generare un certificato e una chiave del client. Durante la generazione, impostare il nome comune (CN) sull'utente ONTAP per l'autenticazione come.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. Aggiungere un certificato CA attendibile al cluster ONTAP. Questo potrebbe essere già gestito dall'amministratore dello storage. Ignorare se non viene utilizzata alcuna CA attendibile.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Installare il certificato e la chiave del client (dal passaggio 1) sul cluster ONTAP.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. Verificare che il ruolo di accesso di sicurezza di ONTAP supporti `cert` metodo di autenticazione.

```
security login create -user-or-group-name admin -application ontapi
-authentication-method cert
security login create -user-or-group-name admin -application http
-authentication-method cert
```

5. Verifica dell'autenticazione utilizzando il certificato generato. Sostituire <LIF di gestione ONTAP> e <vserver name> con IP LIF di gestione e nome SVM.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Codifica certificato, chiave e certificato CA attendibile con Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Creare il backend utilizzando i valori ottenuti dal passaggio precedente.

```
$ cat cert-backend.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaallllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"trustedCACertificate": "QNFinfO...SiqOyN",
"storagePrefix": "myPrefix_"
}

$ tridentctl create backend -f cert-backend.json -n trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                  UUID                  |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       0 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Aggiornare i metodi di autenticazione o ruotare le credenziali

È possibile aggiornare un backend esistente per utilizzare un metodo di autenticazione diverso o per ruotare le credenziali. Questo funziona in entrambi i modi: I backend che utilizzano il nome utente/la password possono essere aggiornati per utilizzare i certificati; i backend che utilizzano i certificati possono essere aggiornati in base al nome utente/alla password. A tale scopo, utilizzare un aggiornamento `backend.json` file contenente i parametri da eseguire `tridentctl backend update`.

```
$ cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "secret",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
$ tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

> Quando si ruotano le password, l'amministratore dello storage deve prima aggiornare la password per l'utente su ONTAP. Seguito da un aggiornamento back-end. Durante la rotazione dei certificati, è possibile aggiungere più certificati all'utente. Il backend viene quindi aggiornato per utilizzare il nuovo certificato, dopodiché il vecchio certificato può essere cancellato dal cluster ONTAP.

L'aggiornamento di un backend non interrompe l'accesso ai volumi già creati, né influisce sulle connessioni dei volumi effettuate successivamente. Un aggiornamento back-end corretto indica che Astra Trident può comunicare con il backend ONTAP e gestire le future operazioni sui volumi.

Specifica igroups

Astra Trident utilizza igroups per controllare l'accesso ai volumi (LUN) forniti. Gli amministratori hanno due opzioni per specificare igroups per i backend:

- Astra Trident può creare e gestire automaticamente un igroup per backend. Se `igroupName` Non è incluso nella definizione di backend, Astra Trident crea un igroup denominato `trident-<backend-UUID>` Su SVM. In questo modo, ciascun backend disporrà di un igroup dedicato e gestirà l'aggiunta/eliminazione automatica degli IQN dei nodi Kubernetes.
- In alternativa, gli igroups pre-creati possono essere forniti anche in una definizione di back-end. Questa

operazione può essere eseguita utilizzando `igroupName` parametro di configurazione. Astra Trident aggiungerà/eliminerà gli IQN dei nodi Kubernetes all'igroup preesistente.

Per i backend che hanno `igroupName` definito, il `igroupName` può essere eliminato con un `tridentctl backend update` Per fare in modo che Astra Trident gestisca automaticamente igroups. In questo modo, l'accesso ai volumi già collegati ai carichi di lavoro non verrà disturbato. Le connessioni future verranno gestite utilizzando il igroup Astra Trident creato.

> Dedicare un igroup per ogni istanza unica di Astra Trident è una Best practice che è vantaggiosa per l'amministratore Kubernetes e per l'amministratore dello storage. CSI Trident automatizza l'aggiunta e la rimozione degli IQN dei nodi del cluster all'igroup, semplificando notevolmente la gestione. Quando si utilizza la stessa SVM in ambienti Kubernetes (e installazioni Astra Trident), l'utilizzo di un igroup dedicato garantisce che le modifiche apportate a un cluster Kubernetes non influiscano sugli igroups associati a un altro. Inoltre, è importante garantire che ciascun nodo del cluster Kubernetes disponga di un IQN univoco. Come indicato in precedenza, Astra Trident gestisce automaticamente l'aggiunta e la rimozione di IQN. Il riutilizzo degli IQN tra gli host può portare a scenari indesiderati in cui gli host si scambiano e l'accesso alle LUN viene negato.

Se Astra Trident è configurato per funzionare come provider CSI, gli IQN dei nodi Kubernetes vengono aggiunti/rimossi automaticamente dall'igroup. Quando i nodi vengono aggiunti a un cluster Kubernetes, `trident-csi` DemonSet implementa un pod (`trident-csi-xxxxx`) sui nodi appena aggiunti e registra i nuovi nodi a cui è possibile collegare i volumi. Gli IQN dei nodi vengono aggiunti anche all'igroup del backend. Un insieme simile di passaggi gestisce la rimozione degli IQN quando i nodi vengono cordonati, scaricati e cancellati da Kubernetes.

Se Astra Trident non viene eseguito come CSI Provisioner, l'igroup deve essere aggiornato manualmente per contenere gli IQN iSCSI di ogni nodo di lavoro nel cluster Kubernetes. Gli IQN dei nodi che fanno parte del cluster Kubernetes dovranno essere aggiunti all'igroup. Analogamente, gli IQN dei nodi rimossi dal cluster Kubernetes devono essere rimossi dall'igroup.

Autenticare le connessioni con CHAP bidirezionale

Astra Trident può autenticare le sessioni iSCSI con CHAP bidirezionale per `ontap-san` e. `ontap-san-economy` driver. Per eseguire questa operazione, è necessario attivare `useCHAP` nella definizione del backend. Quando è impostato su `true`, Astra Trident configura la protezione predefinita dell'iniziatore SVM su CHAP bidirezionale e imposta il nome utente e i segreti del file backend. NetApp consiglia di utilizzare CHAP bidirezionale per autenticare le connessioni. Vedere la seguente configurazione di esempio:

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "FaKePaSsWoRd",
    "igroupName": "trident",
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}
```

> Il `useCHAP` Parameter è un'opzione booleana che può essere configurata una sola volta. L'impostazione predefinita è false. Una volta impostato su true, non è possibile impostarlo su false.

Oltre a `useCHAP=true`, il `chapInitiatorSecret`, `chapTargetInitiatorSecret`, `chapTargetUsername`, e. `chapUsername` i campi devono essere inclusi nella definizione di backend. I segreti possono essere modificati dopo la creazione di un backend mediante l'esecuzione `tridentctl update`.

Come funziona

Per impostazione `useCHAP` A vero, l'amministratore dello storage istruisce Astra Trident a configurare CHAP sul backend dello storage. Ciò include quanto segue:

- Impostazione di CHAP su SVM:
  - Se il tipo di protezione initiator predefinito di SVM è None (impostato per impostazione predefinita) **e** non sono presenti LUN preesistenti nel volume, Astra Trident imposterà il tipo di protezione predefinito su `CHAP` E procedere alla configurazione dell'iniziatore CHAP e del nome utente e dei segreti di destinazione.
  - Se la SVM contiene LUN, Astra Trident non attiverà CHAP sulla SVM. Ciò garantisce che l'accesso alle LUN già presenti sulla SVM non sia limitato.
- Configurazione dell'iniziatore CHAP e del nome utente e dei segreti di destinazione; queste opzioni devono essere specificate nella configurazione del backend (come mostrato sopra).
- Gestione dell'aggiunta di iniziatori a `igroupName` dato nel back-end. Se non specificato, l'impostazione predefinita è `trident`.

Una volta creato il backend, Astra Trident crea un corrispondente `tridentbackend` CRD e memorizza i segreti CHAP e i nomi utente come segreti Kubernetes. Tutti i PVS creati da Astra Trident su questo backend verranno montati e fissati su CHAP.

Ruota le credenziali e aggiorna i back-end

È possibile aggiornare le credenziali CHAP aggiornando i parametri CHAP in `backend.json` file. Per eseguire questa operazione, è necessario aggiornare i segreti CHAP e utilizzare `tridentctl update` per riflettere queste modifiche.

> Quando si aggiornano i segreti CHAP per un backend, è necessario utilizzare `tridentctl` per aggiornare il backend. Non aggiornare le credenziali sul cluster di storage attraverso l'interfaccia utente CLI/ONTAP, in quanto Astra Trident non sarà in grado di rilevare queste modifiche.

```
$ cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "FaKePaSsWoRd",
    "igroupName": "trident",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

$ ./tridentctl update backend ontap_san_chap -f backend-san.json -n
trident
+----------------+----------------+--------------------------------------
+--------+---------+
|   NAME         | STORAGE DRIVER |                  UUID                |
STATE   | VOLUMES |
+----------------+----------------+--------------------------------------
+--------+---------+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |       7 |
+----------------+----------------+--------------------------------------
+--------+---------+
```

Le connessioni esistenti rimarranno inalterate; continueranno a rimanere attive se le credenziali vengono aggiornate da Astra Trident sulla SVM. Le nuove connessioni utilizzeranno le credenziali aggiornate e le connessioni esistenti continueranno a rimanere attive. Disconnettendo e riconnettendo il vecchio PVS, verranno utilizzate le credenziali aggiornate.

Opzioni di configurazione ed esempi

Scopri come creare e utilizzare i driver SAN ONTAP con l'installazione di Astra Trident. Questa sezione fornisce esempi di configurazione back-end e dettagli su come mappare i backend a StorageClasses.

Opzioni di configurazione back-end

Per le opzioni di configurazione del backend, consultare la tabella seguente:

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di storage	"ontap-nas", "ontap-nas-economy", "ontap-nas-flexgroup", "ontap-san", "ontap-san-economy"
`backendName`	Nome personalizzato o backend dello storage	Nome del driver + "_" + dataLIF
`managementLIF`	Indirizzo IP di un cluster o LIF di gestione SVM	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Indirizzo IP del protocollo LIF. USA le parentesi quadre per IPv6. Non può essere aggiornato dopo l'impostazione	Derivato dalla SVM, se non specificato
`useCHAP`	Utilizzare CHAP per autenticare iSCSI per i driver SAN ONTAP [booleano]	falso
`chapInitiatorSecret`	Segreto iniziatore CHAP. Necessario se `useCHAP=true`	""
`labels`	Set di etichette arbitrarie formattate con JSON da applicare sui volumi	""
`chapTargetInitiatorSecret`	CHAP target Initiator secret. Necessario se `useCHAP=true`	""
`chapUsername`	Nome utente inbound. Necessario se `useCHAP=true`	""
`chapTargetUsername`	Nome utente di destinazione. Necessario se `useCHAP=true`	""
`clientCertificate`	Valore del certificato client codificato con base64. Utilizzato per l'autenticazione basata su certificato	""
`clientPrivateKey`	Valore codificato in base64 della chiave privata del client. Utilizzato per l'autenticazione basata su certificato	""

Parametro	Descrizione	Predefinito
`trustedCACertificate`	Valore codificato in base64 del certificato CA attendibile. Opzionale. Utilizzato per l'autenticazione basata su certificato	""
`username`	Nome utente per la connessione al cluster/SVM. Utilizzato per l'autenticazione basata sulle credenziali	""
`password`	Password per la connessione al cluster/SVM. Utilizzato per l'autenticazione basata sulle credenziali	""
`svm`	Macchina virtuale per lo storage da utilizzare	Derivato se un SVM `managementLIF` è specificato
`igroupName`	Nome dell'igroup per i volumi SAN da utilizzare	"Trident-<backend-UUID>"
`storagePrefix`	Prefisso utilizzato per il provisioning di nuovi volumi nella SVM. Non può essere aggiornato dopo l'impostazione	"tridente"
`limitAggregateUsage`	Il provisioning non riesce se l'utilizzo è superiore a questa percentuale. **Non si applica ad Amazon FSX per ONTAP**	"" (non applicato per impostazione predefinita)
`limitVolumeSize`	Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore.	"" (non applicato per impostazione predefinita)
`lunsPerFlexvol`	LUN massimi per FlexVol, devono essere compresi nell'intervallo [50, 200]	"100"
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Ad esempio, {"api":false,} method":true	nullo
`useREST`	Parametro booleano per l'utilizzo delle API REST di ONTAP. **Anteprima tecnica**	falso

> `useREST` viene fornito come **anteprima tecnica** consigliata per ambienti di test e non per carichi di lavoro di produzione. Quando è impostato su `true`, Astra Trident utilizzerà le API REST di ONTAP per comunicare con il backend. Questa funzione richiede ONTAP 9.9 e versioni successive. Inoltre, il ruolo di accesso ONTAP utilizzato deve avere accesso a. `ontap` applicazione. Ciò è soddisfatto dal predefinito `vsadmin` e. `cluster-admin` ruoli.

Per comunicare con il cluster ONTAP, è necessario fornire i parametri di autenticazione. Potrebbe trattarsi del nome utente/password di un account di accesso di sicurezza o di un certificato installato.

> Se si utilizza un backend Amazon FSX per NetApp ONTAP, non specificare `limitAggregateUsage` parametro. Il `fsxadmin` e. `vsadmin` I ruoli forniti da Amazon FSX per NetApp ONTAP non contengono le autorizzazioni di accesso necessarie per recuperare l'utilizzo aggregato e limitarlo tramite Astra Trident.

> Non utilizzare `debugTraceFlags` a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.

Per `ontap-san` Driver, l'impostazione predefinita prevede l'utilizzo di tutti gli IP LIF dei dati dalla SVM e l'utilizzo di multipath iSCSI. Specifica di un indirizzo IP per il dataLIF per `ontap-san` i driver li costringono a disattivare multipath e a utilizzare solo l'indirizzo specificato.

> Quando si crea un backend, ricordarlo `dataLIF` e. `storagePrefix` impossibile modificare dopo la creazione. Per aggiornare questi parametri, è necessario creare un nuovo backend.

`igroupName` Può essere impostato su un igroup già creato nel cluster ONTAP. Se non specificato, Astra Trident crea automaticamente un igroup denominato Trident-<backend-UUID>. Se si fornisce un igroupName predefinito, NetApp consiglia di utilizzare un igroup per cluster Kubernetes, se la SVM deve essere condivisa tra gli ambienti. Ciò è necessario affinché Astra Trident mantenga automaticamente aggiunte/eliminazioni IQN.

I back-end possono anche aggiornare igroups dopo la creazione:

- IgroupName può essere aggiornato per indicare un nuovo igroup creato e gestito sulla SVM all'esterno di Astra Trident.
- IgroupName può essere omesso. In questo caso, Astra Trident creerà e gestirà automaticamente un igroup trident-<backend-UUID>.

In entrambi i casi, gli allegati dei volumi continueranno ad essere accessibili. I futuri allegati dei volumi utilizzeranno l'igroup aggiornato. Questo aggiornamento non interrompe l'accesso ai volumi presenti nel back-end.

È possibile specificare un FQDN (Fully-qualified domain name) per `managementLIF` opzione.

```
`managementLIF` Per tutti i driver ONTAP è possibile impostare anche gli
indirizzi IPv6. Assicurarsi di installare Trident con `--use-ipv6`
allarme. È necessario prestare attenzione alla definizione `managementLIF`
Indirizzo IPv6 tra parentesi quadre.
```

> Quando si utilizzano indirizzi IPv6, assicurarsi `managementLIF` e. `dataLIF` (se incluso nella definizione del backend) sono definiti tra parentesi quadre, ad esempio [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Se `dataLIF` Non è fornito, Astra Trident recupererà i dati IPv6 LIF da SVM.

Per abilitare i driver ontap-san a utilizzare CHAP, impostare `useCHAP` parametro a. `true` nella definizione di back-end. Astra Trident configurerà e utilizzerà CHAP bidirezionale come autenticazione predefinita per la SVM fornita nel backend. Vedere "qui" per scoprire come funziona.

Per `ontap-san-economy` driver, il `limitVolumeSize` L'opzione limita inoltre le dimensioni massime dei volumi gestiti per qtree e LUN.

Astra Trident imposta le etichette di provisioning nel campo "commenti" di tutti i volumi creati utilizzando `ontap-san` driver. Per ogni volume creato, il campo "commenti" di FlexVol contiene tutte le etichette presenti sul pool di storage in cui è inserito. Gli amministratori dello storage possono definire le etichette per ogni pool di storage e raggruppare tutti i volumi creati in un pool di storage. In questo modo è possibile differenziare i volumi in base a una serie di etichette personalizzabili fornite nella configurazione di back-end.

Opzioni di configurazione back-end per il provisioning dei volumi

Per impostazione predefinita, è possibile controllare il provisioning di ciascun volume utilizzando queste opzioni in una sezione speciale della configurazione. Per un esempio, vedere gli esempi di configurazione riportati di seguito.

Parametro	Descrizione	Predefinito
`spaceAllocation`	Allocazione dello spazio per LUN	"vero"
`spaceReserve`	Modalità di riserva dello spazio; "nessuno" (sottile) o "volume" (spesso)	"nessuno"
`snapshotPolicy`	Policy di Snapshot da utilizzare	"nessuno"
`qosPolicy`	Gruppo di criteri QoS da assegnare per i volumi creati. Scegliere tra qosPolicy o adaptiveQosPolicy per pool di storage/backend	""
`adaptiveQosPolicy`	Gruppo di criteri QoS adattivi da assegnare per i volumi creati. Scegliere tra qosPolicy o adaptiveQosPolicy per pool di storage/backend	""
`snapshotReserve`	Percentuale di volume riservato agli snapshot "0"	Se `snapshotPolicy` è "nessuno", altrimenti ""
`splitOnClone`	Separare un clone dal suo padre al momento della creazione	"falso"
`splitOnClone`	Separare un clone dal suo padre al momento della creazione	"falso"
`encryption`	Abilitare la crittografia dei volumi NetApp	"falso"
`securityStyle`	Stile di sicurezza per nuovi volumi	"unix"
`tieringPolicy`	Policy di tiering per utilizzare "nessuno"	"Solo snapshot" per configurazione SVM-DR precedente a ONTAP 9.5

L'utilizzo di gruppi di policy QoS con Astra Trident richiede ONTAP 9.8 o versione successiva. Si consiglia di utilizzare un gruppo di criteri QoS non condiviso e assicurarsi che il gruppo di criteri sia applicato a ciascun componente singolarmente. Un gruppo di policy QoS condiviso applicherà il limite massimo per il throughput totale di tutti i carichi di lavoro.

Ecco un esempio con i valori predefiniti definiti:

```
{
 "version": 1,
 "storageDriverName": "ontap-san",
 "managementLIF": "10.0.0.1",
 "dataLIF": "10.0.0.2",
 "svm": "trident_svm",
 "username": "admin",
 "password": "password",
 "labels": {"k8scluster": "dev2", "backend": "dev2-sanbackend"},
 "storagePrefix": "alternate-trident",
 "igroupName": "custom",
 "debugTraceFlags": {"api":false, "method":true},
 "defaults": {
     "spaceReserve": "volume",
     "qosPolicy": "standard",
     "spaceAllocation": "false",
     "snapshotPolicy": "default",
     "snapshotReserve": "10"
 }
}
```

> Per tutti i volumi creati utilizzando `ontap-san` Driver, Astra Trident aggiunge una capacità extra del 10% a FlexVol per ospitare i metadati LUN. Il LUN viene fornito con le dimensioni esatte richieste dall'utente nel PVC. Astra Trident aggiunge il 10% al FlexVol (viene visualizzato come dimensione disponibile in ONTAP). A questo punto, gli utenti otterranno la quantità di capacità utilizzabile richiesta. Questa modifica impedisce inoltre che le LUN diventino di sola lettura, a meno che lo spazio disponibile non sia completamente utilizzato. Ciò non si applica a ontap-san-Economy.

Per i backend che definiscono `snapshotReserve`, Astra Trident calcola le dimensioni dei volumi come segue:

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
percentage) / 100)] * 1.1
```

Il 1.1 è il 10% aggiuntivo che Astra Trident aggiunge a FlexVol per ospitare i metadati LUN. Per `snapshotReserve` = 5% e richiesta PVC = 5GiB, la dimensione totale del volume è 5,79GiB e la dimensione disponibile è 5,5GiB. Il `volume show` il comando dovrebbe mostrare risultati simili a questo esempio:

```
Vserver   Volume       Aggregate    State      Type       Size  Available Used%
--------- ------------ ------------ ---------- ---- ---------- ---------- -----
          _pvc_89f1c156_3801_4de4_9f9d_034d54c395f4
                                   online     RW         10GB     5.00GB    0%
          _pvc_e42ec6fe_3baa_4af6_996d_134adbbb8e6d
                                   online     RW       5.79GB     5.50GB    0%
          _pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba
                                   online     RW          1GB    511.8MB    0%
3 entries were displayed.
```

Attualmente, il ridimensionamento è l'unico modo per utilizzare il nuovo calcolo per un volume esistente.

Esempi di configurazione minimi

Gli esempi seguenti mostrano le configurazioni di base che lasciano la maggior parte dei parametri predefiniti. Questo è il modo più semplice per definire un backend.

> Se si utilizza Amazon FSX su NetApp ONTAP con Astra Trident, si consiglia di specificare i nomi DNS per i file LIF anziché gli indirizzi IP.

`ontap-san` driver con autenticazione basata su certificato

Si tratta di un esempio minimo di configurazione di back-end. `clientCertificate`, `clientPrivateKey`, e. `trustedCACertificate` (Facoltativo, se si utilizza una CA attendibile) sono inseriti in `backend.json` E prendere rispettivamente i valori codificati base64 del certificato client, della chiave privata e del certificato CA attendibile.

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "DefaultSANBackend",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
    "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
    "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz"
}
```

`ontap-san` Driver con CHAP bidirezionale

Si tratta di un esempio minimo di configurazione di back-end. Questa configurazione di base crea un `ontap-san` back-end con `useCHAP` impostare su `true`.

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "labels": {"k8scluster": "test-cluster-1", "backend": "testcluster1-
sanbackend"},
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "username": "vsadmin",
    "password": "secret"
}
```

`ontap-san-economy` **driver**

```
{
    "version": 1,
    "storageDriverName": "ontap-san-economy",
    "managementLIF": "10.0.0.1",
    "svm": "svm_iscsi_eco",
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "username": "vsadmin",
    "password": "secret"
}
```

Esempi di backend con pool di storage virtuali

Nel file di definizione back-end di esempio mostrato di seguito, vengono impostati valori predefiniti specifici per tutti i pool di storage, ad esempio `spaceReserve` a nessuno, `spaceAllocation` a false, e. `encryption` a falso. I pool di storage virtuali sono definiti nella sezione storage.

In questo esempio, alcuni dei pool di storage vengono impostati in modo personalizzato `spaceReserve`, `spaceAllocation`, e. `encryption` e alcuni pool sovrascrivono i valori predefiniti precedentemente impostati.

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "username": "vsadmin",
    "password": "secret",

    "defaults": {
          "spaceAllocation": "false",
          "encryption": "false",
          "qosPolicy": "standard"
    },
    "labels":{"store": "san_store", "kubernetes-cluster": "prod-cluster-
1"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"protection":"gold", "creditpoints":"40000"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceAllocation": "true",
                "encryption": "true",
                "adaptiveQosPolicy": "adaptive-extreme"
            }
        },
        {
            "labels":{"protection":"silver", "creditpoints":"20000"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceAllocation": "false",
                "encryption": "true",
                "qosPolicy": "premium"
            }
        },
        {
            "labels":{"protection":"bronze", "creditpoints":"5000"},
            "zone":"us_east_1c",
            "defaults": {
```

```
                "spaceAllocation": "true",
                "encryption": "false"
            }
        }
    ]
}
```

Di seguito viene riportato un esempio iSCSI per `ontap-san-economy` driver:

```
{
    "version": 1,
    "storageDriverName": "ontap-san-economy",
    "managementLIF": "10.0.0.1",
    "svm": "svm_iscsi_eco",
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "username": "vsadmin",
    "password": "secret",

    "defaults": {
          "spaceAllocation": "false",
          "encryption": "false"
    },
    "labels":{"store":"san_economy_store"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"app":"oracledb", "cost":"30"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceAllocation": "true",
                "encryption": "true"
            }
        },
        {
            "labels":{"app":"postgresdb", "cost":"20"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceAllocation": "false",
                "encryption": "true"
            }
```

```
        },
        {
            "labels":{"app":"mysqldb", "cost":"10"},
            "zone":"us_east_1c",
            "defaults": {
                "spaceAllocation": "true",
                "encryption": "false"
            }
        }
    ]
}
```

Mappare i backend in StorageClasses

Le seguenti definizioni di StorageClass si riferiscono ai pool di storage virtuali sopra indicati. Utilizzando il `parameters.selector` Ciascun StorageClass richiama i pool virtuali che possono essere utilizzati per ospitare un volume. Gli aspetti del volume saranno definiti nel pool virtuale scelto.

- Il primo StorageClass (`protection-gold`) verrà mappato al primo, secondo pool di storage virtuale in `ontap-nas-flexgroup` il back-end e il primo pool di storage virtuale in `ontap-san` back-end. Si tratta dell'unico pool che offre una protezione di livello gold.
- Il secondo StorageClass (`protection-not-gold`) verrà mappato al terzo e quarto pool di storage virtuale in `ontap-nas-flexgroup` back-end e il secondo, terzo pool di storage virtuale in `ontap-san` back-end. Questi sono gli unici pool che offrono un livello di protezione diverso dall'oro.
- Il terzo StorageClass (`app-mysqldb`) verrà mappato al quarto pool di storage virtuale in `ontap-nas` il back-end e il terzo pool di storage virtuale in `ontap-san-economy` back-end. Questi sono gli unici pool che offrono la configurazione del pool di storage per applicazioni di tipo mysqldb.
- Il quarto StorageClass (`protection-silver-creditpoints-20k`) verrà mappato al terzo pool di storage virtuale in `ontap-nas-flexgroup` il back-end e il secondo pool di storage virtuale in `ontap-san` back-end. Questi sono gli unici pool che offrono una protezione di livello gold a 20000 punti di credito.
- Quinta StorageClass (`creditpoints-5k`) verrà mappato al secondo pool di storage virtuale in `ontap-nas-economy` il back-end e il terzo pool di storage virtuale in `ontap-san` back-end. Queste sono le uniche offerte di pool a 5000 punti di credito.

Astra Trident deciderà quale pool di storage virtuale è selezionato e garantirà il rispetto dei requisiti di storage.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: netapp.io/trident
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Configurare un backend con i driver NAS ONTAP

Informazioni sulla configurazione di un backend ONTAP con driver NAS ONTAP e Cloud Volumes ONTAP.

- "Preparazione"
- "Configurazione ed esempi"

Autorizzazioni utente

Astra Trident prevede di essere eseguito come amministratore di ONTAP o SVM, in genere utilizzando `admin` utente del cluster o un `vsadmin` Utente SVM o un utente con un nome diverso che ha lo stesso ruolo. Per le implementazioni di Amazon FSX per NetApp ONTAP, Astra Trident prevede di essere eseguito come amministratore di ONTAP o SVM, utilizzando il cluster `fsxadmin` utente o a. `vsadmin` Utente SVM o un utente con un nome diverso che ha lo stesso ruolo. Il `fsxadmin` user è un sostituto limitato per l'utente amministratore del cluster.

> Se si utilizza `limitAggregateUsage` parametro, sono richieste le autorizzazioni di amministrazione del cluster. Quando si utilizza Amazon FSX per NetApp ONTAP con Astra Trident, il `limitAggregateUsage` il parametro non funziona con `vsadmin` e. `fsxadmin` account utente. L'operazione di configurazione non riesce se si specifica questo parametro.

Sebbene sia possibile creare un ruolo più restrittivo all'interno di ONTAP che un driver Trident può utilizzare, non lo consigliamo. La maggior parte delle nuove release di Trident chiamerà API aggiuntive che dovrebbero essere considerate, rendendo gli aggiornamenti difficili e soggetti a errori.

Preparazione

Scopri come preparare la configurazione di un backend ONTAP con i driver NAS ONTAP. Per tutti i backend ONTAP, Astra Trident richiede almeno un aggregato assegnato alla SVM.

Per tutti i backend ONTAP, Astra Trident richiede almeno un aggregato assegnato alla SVM.

È inoltre possibile eseguire più di un driver e creare classi di storage che puntino all'una o all'altra. Ad esempio, è possibile configurare una classe Gold che utilizza `ontap-nas` Driver e una classe Bronze che utilizza `ontap-nas-economy` uno.

Tutti i nodi di lavoro di Kubernetes devono avere installati gli strumenti NFS appropriati. Vedere "qui" per ulteriori dettagli.

Autenticazione

Astra Trident offre due modalità di autenticazione di un backend ONTAP.

- Basato sulle credenziali: Nome utente e password di un utente ONTAP con le autorizzazioni richieste. Si consiglia di utilizzare un ruolo di accesso di sicurezza predefinito, ad esempio `admin` oppure `vsadmin` Per garantire la massima compatibilità con le versioni di ONTAP.
- Basato su certificato: Astra Trident può anche comunicare con un cluster ONTAP utilizzando un certificato installato sul backend. In questo caso, la definizione di backend deve contenere i valori codificati in Base64 del certificato client, della chiave e del certificato CA attendibile, se utilizzato (consigliato).

Gli utenti possono anche scegliere di aggiornare i back-end esistenti, scegliendo di passare da basato su credenziali a basato su certificato e viceversa. Se **vengono forniti sia credenziali che certificati**, Astra Trident utilizza per impostazione predefinita i certificati mentre emette un avviso per rimuovere le credenziali

dalla definizione di backend.

Abilitare l'autenticazione basata su credenziali

Astra Trident richiede le credenziali di un amministratore con ambito SVM/cluster per comunicare con il backend ONTAP. Si consiglia di utilizzare ruoli standard predefiniti, ad esempio `admin` oppure `vsadmin`. Ciò garantisce la compatibilità con le future release di ONTAP che potrebbero esporre le API delle funzionalità da utilizzare nelle future release di Astra Trident. È possibile creare e utilizzare un ruolo di accesso di sicurezza personalizzato con Astra Trident, ma non è consigliato.

Una definizione di back-end di esempio avrà un aspetto simile al seguente:

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "secret"
}
```

Tenere presente che la definizione di backend è l'unica posizione in cui le credenziali vengono memorizzate in testo normale. Una volta creato il backend, i nomi utente e le password vengono codificati con Base64 e memorizzati come segreti Kubernetes. La creazione/l'updation di un backend è l'unico passaggio che richiede la conoscenza delle credenziali. Pertanto, si tratta di un'operazione di sola amministrazione, che deve essere eseguita dall'amministratore Kubernetes/storage.

Abilitare l'autenticazione basata su certificato

I backend nuovi ed esistenti possono utilizzare un certificato e comunicare con il backend ONTAP. Nella definizione di backend sono necessari tre parametri.

- ClientCertificate: Valore del certificato client codificato con base64.
- ClientPrivateKey: Valore codificato in base64 della chiave privata associata.
- TrustedCACertificate: Valore codificato in base64 del certificato CA attendibile. Se si utilizza una CA attendibile, è necessario fornire questo parametro. Questa operazione può essere ignorata se non viene utilizzata alcuna CA attendibile.

Un workflow tipico prevede i seguenti passaggi.

Fasi

1. Generare un certificato e una chiave del client. Durante la generazione, impostare il nome comune (CN) sull'utente ONTAP per l'autenticazione come.

   ```
   openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
   -out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
   ```

2. Aggiungere un certificato CA attendibile al cluster ONTAP. Questo potrebbe essere già gestito dall'amministratore dello storage. Ignorare se non viene utilizzata alcuna CA attendibile.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Installare il certificato e la chiave del client (dal passaggio 1) sul cluster ONTAP.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. Verificare che il ruolo di accesso di sicurezza di ONTAP supporti `cert` metodo di autenticazione.

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

5. Verifica dell'autenticazione utilizzando il certificato generato. Sostituire <LIF di gestione ONTAP> e <vserver name> con IP LIF di gestione e nome SVM. Assicurarsi che la politica di servizio di LIF sia impostata su `default-data-management`.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Codifica certificato, chiave e certificato CA attendibile con Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Creare il backend utilizzando i valori ottenuti dal passaggio precedente.

```
$ cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
$ tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Aggiornare i metodi di autenticazione o ruotare le credenziali

È possibile aggiornare un backend esistente per utilizzare un metodo di autenticazione diverso o per ruotare le credenziali. Questo funziona in entrambi i modi: I backend che utilizzano il nome utente/la password possono essere aggiornati per utilizzare i certificati; i backend che utilizzano i certificati possono essere aggiornati in base al nome utente/alla password. A tale scopo, utilizzare un aggiornamento `backend.json` file contenente i parametri da eseguire `tridentctl backend update`.

```
$ cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "secret",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
$ tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

> Quando si ruotano le password, l'amministratore dello storage deve prima aggiornare la password per l'utente su ONTAP. Seguito da un aggiornamento back-end. Durante la rotazione dei certificati, è possibile aggiungere più certificati all'utente. Il backend viene quindi aggiornato per utilizzare il nuovo certificato, dopodiché il vecchio certificato può essere cancellato dal cluster ONTAP.

L'aggiornamento di un backend non interrompe l'accesso ai volumi già creati, né influisce sulle connessioni dei volumi effettuate successivamente. Un aggiornamento back-end corretto indica che Astra Trident può comunicare con il backend ONTAP e gestire le future operazioni sui volumi.

Gestire le policy di esportazione NFS

Astra Trident utilizza policy di esportazione NFS per controllare l'accesso ai volumi forniti dall'IT.

Astra Trident offre due opzioni quando si lavora con le policy di esportazione:

- Astra Trident è in grado di gestire dinamicamente la policy di esportazione; in questa modalità operativa, l'amministratore dello storage specifica un elenco di blocchi CIDR che rappresentano indirizzi IP consentiti. Astra Trident aggiunge automaticamente gli IP dei nodi che rientrano in questi intervalli ai criteri di esportazione. In alternativa, se non viene specificato alcun CIDR, qualsiasi IP unicast con ambito globale trovato nei nodi verrà aggiunto alla policy di esportazione.

- Gli amministratori dello storage possono creare una policy di esportazione e aggiungere regole manualmente. Astra Trident utilizza il criterio di esportazione predefinito, a meno che nella configurazione non venga specificato un nome diverso del criterio di esportazione.

Gestione dinamica delle policy di esportazione

La versione 20.04 di CSI Trident offre la possibilità di gestire dinamicamente le policy di esportazione per i backend ONTAP. In questo modo, l'amministratore dello storage può specificare uno spazio di indirizzi consentito per gli IP dei nodi di lavoro, invece di definire manualmente regole esplicite. Semplifica notevolmente la gestione delle policy di esportazione; le modifiche alle policy di esportazione non richiedono più l'intervento manuale sul cluster di storage. Inoltre, questo consente di limitare l'accesso al cluster di storage solo ai nodi di lavoro che hanno IP nell'intervallo specificato, supportando una gestione dettagliata e automatica.

> La gestione dinamica delle policy di esportazione è disponibile solo per CSI Trident. È importante assicurarsi che i nodi di lavoro non vengano sottoposti a NATing.

Esempio

È necessario utilizzare due opzioni di configurazione. Ecco un esempio di definizione back-end:

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "ontap_nas_auto_export,
    "managementLIF": "192.168.0.135",
    "svm": "svm1",
    "username": "vsadmin",
    "password": "FaKePaSsWoRd",
    "autoExportCIDRs": ["192.168.0.0/24"],
    "autoExportPolicy": true
}
```

> Quando si utilizza questa funzione, è necessario assicurarsi che la giunzione root di SVM disponga di un criterio di esportazione precreato con una regola di esportazione che consenta il blocco CIDR del nodo (ad esempio il criterio di esportazione predefinito). Seguire sempre le Best practice consigliate da NetApp per dedicare una SVM ad Astra Trident.

Ecco una spiegazione del funzionamento di questa funzione utilizzando l'esempio precedente:

- `autoExportPolicy` è impostato su `true`. Questo indica che Astra Trident creerà un criterio di esportazione per `svm1` SVM e gestire l'aggiunta e l'eliminazione di regole utilizzando `autoExportCIDRs` blocchi di indirizzi. Ad esempio, un backend con UUID 403b5326-8482-40db-96d0-d83fb3f4daec e. `autoExportPolicy` impostare su `true` crea un criterio di esportazione denominato `trident-403b5326-8482-40db-96d0-d83fb3f4daec` Su SVM.
- `autoExportCIDRs` contiene un elenco di blocchi di indirizzi. Questo campo è opzionale e per impostazione predefinita è ["0.0.0.0/0", "::/0"]. Se non definito, Astra Trident aggiunge tutti gli indirizzi unicast con ambito globale trovati nei nodi di lavoro.

In questo esempio, il `192.168.0.0/24` viene fornito uno spazio per gli indirizzi. Ciò indica che gli IP dei nodi Kubernetes che rientrano in questo intervallo di indirizzi verranno aggiunti alla policy di esportazione creata da Astra Trident. Quando Astra Trident registra un nodo su cui viene eseguito, recupera gli indirizzi IP del nodo e li confronta con i blocchi di indirizzo forniti in `autoExportCIDRs`. Dopo aver filtrato gli IP, Astra Trident crea regole di policy di esportazione per gli IP client individuati, con una regola per ogni nodo identificato.

È possibile eseguire l'aggiornamento `autoExportPolicy` e. `autoExportCIDRs` per i backend dopo la creazione. È possibile aggiungere nuovi CIDR a un backend gestito automaticamente o eliminare i CIDR esistenti. Prestare attenzione quando si eliminano i CIDR per assicurarsi che le connessioni esistenti non vengano interrotte. È anche possibile scegliere di disattivare `autoExportPolicy` per un backend e tornare a una policy di esportazione creata manualmente. Questa operazione richiede l'impostazione di `exportPolicy` nella configurazione del backend.

Dopo che Astra Trident ha creato o aggiornato un backend, è possibile controllare il backend utilizzando `tridentctl` o il corrispondente `tridentbackend` CRD:

```
$ ./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileSystemType: ext4
```

Quando i nodi vengono aggiunti a un cluster Kubernetes e registrati con il controller Astra Trident, le policy di esportazione dei backend esistenti vengono aggiornate (a condizione che rientrino nell'intervallo di indirizzi specificato nella `autoExportCIDRs` per il back-end).

Quando un nodo viene rimosso, Astra Trident controlla tutti i backend in linea per rimuovere la regola di accesso per il nodo. Rimuovendo questo IP del nodo dalle policy di esportazione dei backend gestiti, Astra Trident impedisce i montaggi non autorizzati, a meno che questo IP non venga riutilizzato da un nuovo nodo nel cluster.

Per i backend esistenti in precedenza, aggiornare il backend con `tridentctl update backend` Garantisce che Astra Trident gestisca automaticamente le policy di esportazione. In questo modo si crea una nuova policy di esportazione denominata dopo l'UUID del backend e i volumi presenti sul backend utilizzeranno la policy di

esportazione appena creata una volta rimontati.

> L'eliminazione di un backend con policy di esportazione gestite automaticamente elimina la policy di esportazione creata dinamicamente. Se il backend viene ricreato, viene trattato come un nuovo backend e si otterrà la creazione di una nuova policy di esportazione.

Se l'indirizzo IP di un nodo live viene aggiornato, è necessario riavviare il pod Astra Trident sul nodo. Astra Trident aggiornerà quindi la policy di esportazione per i backend che riesce a riflettere questa modifica IP.

Opzioni di configurazione ed esempi

Scopri come creare e utilizzare i driver NAS ONTAP con l'installazione di Astra Trident. Questa sezione fornisce esempi di configurazione back-end e dettagli su come mappare i backend a StorageClasses.

Opzioni di configurazione back-end

Per le opzioni di configurazione del backend, consultare la tabella seguente:

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di storage	"ontap-nas", "ontap-nas-economy", "ontap-nas-flexgroup", "ontap-san", "ontap-san-economy"
`backendName`	Nome personalizzato o backend dello storage	Nome del driver + "_" + dataLIF
`managementLIF`	Indirizzo IP di un cluster o LIF di gestione SVM	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Indirizzo IP del protocollo LIF. USA le parentesi quadre per IPv6. Non può essere aggiornato dopo l'impostazione	Derivato dalla SVM, se non specificato
`autoExportPolicy`	Abilitare la creazione e l'aggiornamento automatici dei criteri di esportazione [booleano]	falso
`autoExportCIDRs`	Elenco di CIDR per filtrare gli IP dei nodi Kubernetes rispetto a quando `autoExportPolicy` è attivato	["0.0.0.0/0", ":/0"]`
`labels`	Set di etichette arbitrarie formattate con JSON da applicare sui volumi	""
`clientCertificate`	Valore del certificato client codificato con base64. Utilizzato per l'autenticazione basata su certificato	""
`clientPrivateKey`	Valore codificato in base64 della chiave privata del client. Utilizzato per l'autenticazione basata su certificato	""

Parametro	Descrizione	Predefinito
`trustedCACertificate`	Valore codificato in base64 del certificato CA attendibile. Opzionale. Utilizzato per l'autenticazione basata su certificato	""
`username`	Nome utente per la connessione al cluster/SVM. Utilizzato per l'autenticazione basata sulle credenziali	
`password`	Password per la connessione al cluster/SVM. Utilizzato per l'autenticazione basata sulle credenziali	
`svm`	Macchina virtuale per lo storage da utilizzare	Derivato se un SVM `managementLIF` è specificato
`igroupName`	Nome dell'igroup per i volumi SAN da utilizzare	"Trident-<backend-UUID>"
`storagePrefix`	Prefisso utilizzato per il provisioning di nuovi volumi nella SVM. Non può essere aggiornato dopo l'impostazione	"tridente"
`limitAggregateUsage`	Il provisioning non riesce se l'utilizzo è superiore a questa percentuale. **Non si applica ad Amazon FSX per ONTAP**	"" (non applicato per impostazione predefinita)
`limitVolumeSize`	Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore.	"" (non applicato per impostazione predefinita)
`lunsPerFlexvol`	LUN massimi per FlexVol, devono essere compresi nell'intervallo [50, 200]	"100"
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Ad esempio, {"api":false,} method":true	nullo
`nfsMountOptions`	Elenco separato da virgole delle opzioni di montaggio NFS	""
`qtreesPerFlexvol`	Qtree massimi per FlexVol, devono essere compresi nell'intervallo [50, 300]	"200"
`useREST`	Parametro booleano per l'utilizzo delle API REST di ONTAP. **Anteprima tecnica**	falso

> `useREST` viene fornito come **anteprima tecnica** consigliata per ambienti di test e non per carichi di lavoro di produzione. Quando è impostato su `true`, Astra Trident utilizzerà le API REST di ONTAP per comunicare con il backend. Questa funzione richiede ONTAP 9.9 e versioni successive. Inoltre, il ruolo di accesso ONTAP utilizzato deve avere accesso a. `ontap` applicazione. Ciò è soddisfatto dal predefinito `vsadmin` e. `cluster-admin` ruoli.

Per comunicare con il cluster ONTAP, è necessario fornire i parametri di autenticazione. Potrebbe trattarsi del nome utente/password di un account di accesso di sicurezza o di un certificato installato.

> Se si utilizza un backend Amazon FSX per NetApp ONTAP, non specificare `limitAggregateUsage` parametro. Il `fsxadmin` e. `vsadmin` I ruoli forniti da Amazon FSX per NetApp ONTAP non contengono le autorizzazioni di accesso necessarie per recuperare l'utilizzo aggregato e limitarlo tramite Astra Trident.

> Non utilizzare `debugTraceFlags` a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.

> Quando si crea un backend, tenere presente che il `dataLIF` e. `storagePrefix` impossibile modificare dopo la creazione. Per aggiornare questi parametri, è necessario creare un nuovo backend.

È possibile specificare un FQDN (Fully-qualified domain name) per `managementLIF` opzione. È inoltre possibile specificare un FQDN per `dataLIF` In questo caso, l'FQDN verrà utilizzato per le operazioni di montaggio NFS. In questo modo è possibile creare un DNS round-robin per il bilanciamento del carico tra più LIF di dati.

```
`managementLIF` Per tutti i driver ONTAP è possibile impostare anche gli
indirizzi IPv6. Assicurarsi di installare Astra Trident con `--use-ipv6`
allarme. È necessario prestare attenzione alla definizione di
`managementLIF` Indirizzo IPv6 tra parentesi quadre.
```

> Quando si utilizzano indirizzi IPv6, assicurarsi `managementLIF` e. `dataLIF` (se incluso nella definizione del backend) sono definiti tra parentesi quadre, ad esempio [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Se `dataLIF` Non è fornito, Astra Trident recupererà i dati IPv6 LIF da SVM.

Utilizzando il `autoExportPolicy` e. `autoExportCIDRs` CSI Trident è in grado di gestire automaticamente le policy di esportazione. Questo è supportato per tutti i driver ontap-nas-*.

Per `ontap-nas-economy` driver, il `limitVolumeSize` L'opzione limita inoltre le dimensioni massime dei volumi gestiti per qtree e LUN e l' `qtreesPerFlexvol` Consente di personalizzare il numero massimo di qtree per FlexVol.

Il `nfsMountOptions` il parametro può essere utilizzato per specificare le opzioni di montaggio. Le opzioni di montaggio per i volumi persistenti di Kubernetes sono normalmente specificate nelle classi di storage, ma se non sono specificate opzioni di montaggio in una classe di storage, Astra Trident tornerà a utilizzare le opzioni di montaggio specificate nel file di configurazione del backend di storage. Se non sono specificate opzioni di montaggio nella classe di storage o nel file di configurazione, Astra Trident non imposta alcuna opzione di montaggio su un volume persistente associato.

Astra Trident imposta le etichette di provisioning nel campo "commenti" di tutti i volumi creati con(`ontap-nas` e.(`ontap-nas-flexgroup`. In base al driver utilizzato, i commenti vengono impostati su FlexVol (`ontap-nas`) O FlexGroup (`ontap-nas-flexgroup`). Astra Trident copia tutte le etichette presenti in un pool di storage nel volume di storage al momento del provisioning. Gli amministratori dello storage possono definire le etichette per ogni pool di storage e raggruppare tutti i volumi creati in un pool di storage. In questo modo è possibile differenziare i volumi in base a una serie di etichette personalizzabili fornite nella configurazione di back-end.

Opzioni di configurazione back-end per il provisioning dei volumi

Per impostazione predefinita, è possibile controllare il provisioning di ciascun volume utilizzando queste opzioni in una sezione speciale della configurazione. Per un esempio, vedere gli esempi di configurazione riportati di seguito.

Parametro	Descrizione	Predefinito
`spaceAllocation`	Allocazione dello spazio per LUN	"vero"
`spaceReserve`	Modalità di riserva dello spazio; "nessuno" (sottile) o "volume" (spesso)	"nessuno"
`snapshotPolicy`	Policy di Snapshot da utilizzare	"nessuno"
`qosPolicy`	Gruppo di criteri QoS da assegnare per i volumi creati. Scegliere tra qosPolicy o adaptiveQosPolicy per pool di storage/backend	""
`adaptiveQosPolicy`	Gruppo di criteri QoS adattivi da assegnare per i volumi creati. Scegliere tra qosPolicy o adaptiveQosPolicy per pool di storage/backend. Non supportato da ontap-nas-Economy.	""
`snapshotReserve`	Percentuale di volume riservato agli snapshot "0"	Se `snapshotPolicy` è "nessuno", altrimenti ""
`splitOnClone`	Separare un clone dal suo padre al momento della creazione	"falso"
`encryption`	Abilitare la crittografia dei volumi NetApp	"falso"
`securityStyle`	Stile di sicurezza per nuovi volumi	"unix"
`tieringPolicy`	Policy di tiering per utilizzare "nessuno"	"Solo snapshot" per configurazione SVM-DR precedente a ONTAP 9.5
UnixPermissions	Per i nuovi volumi	"777"
SnapshotDir	Controlla la visibilità di `.snapshot` directory	"falso"
ExportPolicy	Policy di esportazione da utilizzare	"predefinito"
SecurityStyle	Stile di sicurezza per nuovi volumi	"unix"

L'utilizzo di gruppi di policy QoS con Astra Trident richiede ONTAP 9.8 o versione successiva. Si consiglia di utilizzare un gruppo di criteri QoS non condiviso e assicurarsi che il gruppo di criteri sia applicato a ciascun componente singolarmente. Un gruppo di policy QoS condiviso applicherà il limite massimo per il throughput totale di tutti i carichi di lavoro.

Ecco un esempio con i valori predefiniti definiti:

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "customBackendName",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "labels": {"k8scluster": "dev1", "backend": "dev1-nasbackend"},
  "svm": "trident_svm",
  "username": "cluster-admin",
  "password": "password",
  "limitAggregateUsage": "80%",
  "limitVolumeSize": "50Gi",
  "nfsMountOptions": "nfsvers=4",
  "debugTraceFlags": {"api":false, "method":true},
  "defaults": {
    "spaceReserve": "volume",
    "qosPolicy": "premium",
    "exportPolicy": "myk8scluster",
    "snapshotPolicy": "default",
    "snapshotReserve": "10"
  }
}
```

Per `ontap-nas` e. `ontap-nas-flexgroups`, Astra Trident utilizza ora un nuovo calcolo per garantire che il FlexVol sia dimensionato correttamente con la percentuale di snapshotReserve e PVC. Quando l'utente richiede un PVC, Astra Trident crea il FlexVol originale con più spazio utilizzando il nuovo calcolo. Questo calcolo garantisce che l'utente riceva lo spazio scrivibile richiesto nel PVC e non uno spazio inferiore a quello richiesto. Prima della versione 21.07, quando l'utente richiede un PVC (ad esempio, 5GiB), con SnapshotReserve al 50%, ottiene solo 2,5 GiB di spazio scrivibile. Questo perché ciò che l'utente ha richiesto è l'intero volume e. `snapshotReserve` è una percentuale. Con Trident 21.07, ciò che l'utente richiede è lo spazio scrivibile e Astra Trident definisce `snapshotReserve` numero come percentuale dell'intero volume. Questo non si applica a. `ontap-nas-economy`. Vedere l'esempio seguente per vedere come funziona:

Il calcolo è il seguente:

```
Total volume size = (PVC requested size) / (1 - (snapshotReserve
percentage) / 100)
```

Per snapshotReserve = 50% e richiesta PVC = 5GiB, la dimensione totale del volume è 2/0,5 = 10GiB e la

dimensione disponibile è 5GiB, che è ciò che l'utente ha richiesto nella richiesta PVC. Il `volume show` il comando dovrebbe mostrare risultati simili a questo esempio:

```
Vserver   Volume       Aggregate    State      Type       Size  Available Used%
--------- ------------ ------------ ---------- ---- ---------- ---------- -----
                 _pvc_89f1c156_3801_4de4_9f9d_034d54c395f4
                                      online     RW         10GB     5.00GB    0%
                 _pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba
                                      online     RW          1GB    511.8MB    0%
2 entries were displayed.
```

I backend esistenti delle installazioni precedenti eseguiranno il provisioning dei volumi come spiegato in precedenza durante l'aggiornamento di Astra Trident. Per i volumi creati prima dell'aggiornamento, è necessario ridimensionare i volumi per osservare la modifica. Ad esempio, un PVC 2GiB con `snapshotReserve=50` In precedenza, si è creato un volume che fornisce 1 GB di spazio scrivibile. Il ridimensionamento del volume su 3GiB, ad esempio, fornisce all'applicazione 3GiB di spazio scrivibile su un volume da 6 GiB.

Esempi di configurazione minimi

Gli esempi seguenti mostrano le configurazioni di base che lasciano la maggior parte dei parametri predefiniti. Questo è il modo più semplice per definire un backend.

> Se si utilizza Amazon FSX su NetApp ONTAP con Trident, si consiglia di specificare i nomi DNS per le LIF anziché gli indirizzi IP.

`ontap-nas` driver con autenticazione basata su certificato

Si tratta di un esempio minimo di configurazione di back-end. `clientCertificate`, `clientPrivateKey`, e. `trustedCACertificate` (Facoltativo, se si utilizza una CA attendibile) sono inseriti in `backend.json` E prendere rispettivamente i valori codificati base64 del certificato client, della chiave privata e del certificato CA attendibile.

```
{
  "version": 1,
  "backendName": "DefaultNASBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.15",
  "svm": "nfs_svm",
  "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
  "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
  "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz",
  "storagePrefix": "myPrefix_"
}
```

`ontap-nas` driver con policy di esportazione automatica

Questo esempio mostra come impostare Astra Trident a utilizzare policy di esportazione dinamiche per creare e gestire automaticamente le policy di esportazione. Questo funziona allo stesso modo per `ontap-nas-`

`economy` e. `ontap-nas-flexgroup` driver.

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "labels": {"k8scluster": "test-cluster-east-1a", "backend": "test1-
nasbackend"},
    "autoExportPolicy": true,
    "autoExportCIDRs": ["10.0.0.0/24"],
    "username": "admin",
    "password": "secret",
    "nfsMountOptions": "nfsvers=4",
}
```

`ontap-nas-flexgroup` **driver**

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "labels": {"k8scluster": "test-cluster-east-1b", "backend": "test1-
ontap-cluster"},
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",
}
```

`ontap-nas` **Driver con IPv6**

```
{
 "version": 1,
 "storageDriverName": "ontap-nas",
 "backendName": "nas_ipv6_backend",
 "managementLIF": "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]",
 "labels": {"k8scluster": "test-cluster-east-1a", "backend": "test1-ontap-
ipv6"},
 "svm": "nas_ipv6_svm",
 "username": "vsadmin",
 "password": "netapp123"
}
```

`ontap-nas-economy` **driver**

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret"
}
```

Esempi di backend con pool di storage virtuali

Nel file di definizione back-end di esempio mostrato di seguito, vengono impostati valori predefiniti specifici per tutti i pool di storage, ad esempio `spaceReserve` a nessuno, `spaceAllocation` a false, e. `encryption` a falso. I pool di storage virtuali sono definiti nella sezione storage.

In questo esempio, alcuni dei pool di storage vengono impostati in modo personalizzato `spaceReserve`, `spaceAllocation`, e. `encryption` e alcuni pool sovrascrivono i valori predefiniti precedentemente impostati.

`ontap-nas` **driver**

```
{
    {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "admin",
    "password": "secret",
```

```
    "nfsMountOptions": "nfsvers=4",

    "defaults": {
          "spaceReserve": "none",
          "encryption": "false",
          "qosPolicy": "standard"
    },
    "labels":{"store":"nas_store", "k8scluster": "prod-cluster-1"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"app":"msoffice", "cost":"100"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755",
                "adaptiveQosPolicy": "adaptive-premium"
            }
        },
        {
            "labels":{"app":"slack", "cost":"75"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"app":"wordpress", "cost":"50"},
            "zone":"us_east_1c",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0775"
            }
        },
        {
            "labels":{"app":"mysqldb", "cost":"25"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
```

```
        }
    ]
}
```

`ontap-nas-flexgroup` **driver**

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",

    "defaults": {
          "spaceReserve": "none",
          "encryption": "false"
    },
    "labels":{"store":"flexgroup_store", "k8scluster": "prod-cluster-1"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"protection":"gold", "creditpoints":"50000"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"protection":"gold", "creditpoints":"30000"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"protection":"silver", "creditpoints":"20000"},
            "zone":"us_east_1c",
            "defaults": {
                "spaceReserve": "none",
```

```
                "encryption": "true",
                "unixPermissions": "0775"
            }
        },
        {
            "labels":{"protection":"bronze", "creditpoints":"10000"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
        }
    ]
}
```

`ontap-nas-economy` **driver**

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",

    "defaults": {
          "spaceReserve": "none",
          "encryption": "false"
    },
    "labels":{"store":"nas_economy_store"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"department":"finance", "creditpoints":"6000"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"department":"legal", "creditpoints":"5000"},
```

```
            "zone":"us_east_1b",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"department":"engineering", "creditpoints":"3000"},
            "zone":"us_east_1c",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0775"
            }
        },
        {
            "labels":{"department":"humanresource",
"creditpoints":"2000"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
        }
    ]
}
```

Mappare i backend in StorageClasses

Le seguenti definizioni di StorageClass si riferiscono ai pool di storage virtuali sopra indicati. Utilizzando il `parameters.selector` Ciascun StorageClass richiama i pool virtuali che possono essere utilizzati per ospitare un volume. Gli aspetti del volume saranno definiti nel pool virtuale scelto.

- Il primo StorageClass (`protection-gold`) verrà mappato al primo, secondo pool di storage virtuale in `ontap-nas-flexgroup` il back-end e il primo pool di storage virtuale in `ontap-san` back-end. Si tratta dell'unico pool che offre una protezione di livello gold.
- Il secondo StorageClass (`protection-not-gold`) verrà mappato al terzo e quarto pool di storage virtuale in `ontap-nas-flexgroup` back-end e il secondo, terzo pool di storage virtuale in `ontap-san` back-end. Questi sono gli unici pool che offrono un livello di protezione diverso dall'oro.
- Il terzo StorageClass (`app-mysqldb`) verrà mappato al quarto pool di storage virtuale in `ontap-nas` il back-end e il terzo pool di storage virtuale in `ontap-san-economy` back-end. Questi sono gli unici pool che offrono la configurazione del pool di storage per applicazioni di tipo mysqldb.
- Il quarto StorageClass (`protection-silver-creditpoints-20k`) verrà mappato al terzo pool di storage virtuale in `ontap-nas-flexgroup` il back-end e il secondo pool di storage virtuale in `ontap-san` back-end. Questi sono gli unici pool che offrono una protezione di livello gold a 20000 punti di credito.

- Quinta StorageClass (`creditpoints-5k`) verrà mappato al secondo pool di storage virtuale in `ontap-nas-economy` il back-end e il terzo pool di storage virtuale in `ontap-san` back-end. Queste sono le uniche offerte di pool a 5000 punti di credito.

Astra Trident deciderà quale pool di storage virtuale è selezionato e garantirà il rispetto dei requisiti di storage.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: netapp.io/trident
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

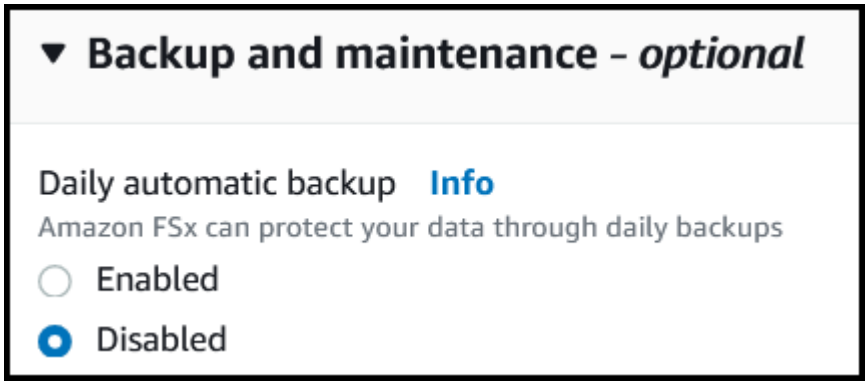
Utilizza Astra Trident con Amazon FSX per NetApp ONTAP

"Amazon FSX per NetApp ONTAP", È un servizio AWS completamente gestito che consente ai clienti di lanciare ed eseguire file system basati sul sistema operativo per lo storage ONTAP di NetApp. Amazon FSX per NetApp ONTAP ti consente di sfruttare le funzionalità, le performance e le funzionalità amministrative di NetApp che conosci, sfruttando al contempo la semplicità, l'agilità, la sicurezza e la scalabilità dell'archiviazione dei dati su AWS. FSX supporta molte delle funzionalità del file system e delle API di amministrazione di ONTAP.

Un file system è la risorsa principale di Amazon FSX, simile a un cluster ONTAP on-premise. All'interno di ogni SVM è possibile creare uno o più volumi, ovvero contenitori di dati che memorizzano i file e le cartelle nel file system. Con Amazon FSX per NetApp ONTAP, Data ONTAP verrà fornito come file system gestito nel cloud. Il nuovo tipo di file system è denominato **NetApp ONTAP**.

Utilizzando Astra Trident con Amazon FSX per NetApp ONTAP, puoi garantire che i cluster Kubernetes in esecuzione in Amazon Elastic Kubernetes Service (EKS) possano eseguire il provisioning di volumi persistenti di file e blocchi supportati da ONTAP.

> **Creazione del file system Amazon FSX per ONTAP**
>
> I volumi creati su file system Amazon FSX con backup automatici attivati non possono essere cancellati da Trident. Per eliminare i PVC, è necessario eliminare manualmente il volume FV e FSX per ONTAP.
>
> Per evitare questo problema:
>
> - Non utilizzare **creazione rapida** per creare il file system FSX per ONTAP. Il workflow di creazione rapida consente backup automatici e non offre un'opzione di opt-out.
> - Quando si utilizza **creazione standard**, disattivare il backup automatico. La disattivazione dei backup automatici consente a Trident di eliminare un volume senza ulteriori interventi manuali.



Scopri Astra Trident

Se sei un nuovo utente di Astra Trident, puoi familiarizzare con i link riportati di seguito:

- "FAQ"
- "Requisiti per l'utilizzo di Astra Trident"
- "Implementare Astra Trident"
- "Best practice per la configurazione di ONTAP, Cloud Volumes ONTAP e Amazon FSX per NetApp ONTAP"

- "Integrare Astra Trident"
- "Configurazione backend SAN ONTAP"
- "Configurazione backend NAS ONTAP"

Scopri di più sulle funzionalità dei driver "qui".

Amazon FSX per NetApp ONTAP utilizza "FabricPool" per gestire i tier di storage. Consente di memorizzare i dati in un Tier, in base all'accesso frequente ai dati.

Astra Trident prevede di essere eseguito come a. `vsadmin` Utente SVM o come utente con un nome diverso che ha lo stesso ruolo. Amazon FSX per NetApp ONTAP ha un `fsxadmin` Utente che sostituisce in maniera limitata il ONTAP `admin` utente del cluster. Si sconsiglia di utilizzare `fsxadmin` Utente, con Trident, come a. `vsadmin` L'utente di SVM ha accesso a più funzionalità di Astra Trident.

Driver

Puoi integrare Astra Trident con Amazon FSX per NetApp ONTAP utilizzando i seguenti driver:

- `ontap-san`: Ogni PV fornito è un LUN all'interno del proprio volume Amazon FSX per NetApp ONTAP.
- `ontap-san-economy`: Ogni PV fornito è un LUN con un numero configurabile di LUN per volume Amazon FSX per NetApp ONTAP.
- `ontap-nas`: Ogni PV fornito è un volume Amazon FSX completo per NetApp ONTAP.
- `ontap-nas-economy`: Ogni PV fornito è un qtree, con un numero configurabile di qtree per ogni volume Amazon FSX per NetApp ONTAP.
- `ontap-nas-flexgroup`: Ogni PV fornito è un volume Amazon FSX completo per NetApp ONTAP FlexGroup.

Autenticazione

Astra Trident offre due modalità di autenticazione:

- Basato su certificato: Astra Trident comunicherà con SVM sul file system FSX utilizzando un certificato installato sulla SVM.
- Basato sulle credenziali: È possibile utilizzare `fsxadmin` utente per il file system o l' `vsadmin` Configurato dall'utente per la SVM.

> Si consiglia vivamente di utilizzare `vsadmin` al posto di `fsxadmin` per configurare il backend. Astra Trident comunicherà con il file system FSX utilizzando questo nome utente e la password.

Per ulteriori informazioni sull'autenticazione, consulta i seguenti "NAS ONTAP"
* "ONTAP SAN"

Implementa e configura Astra Trident su EKS con Amazon FSX per NetApp ONTAP

Di cosa hai bisogno

- Un cluster Amazon EKS esistente o un cluster Kubernetes autogestito con `kubectl` installato.
- Una macchina virtuale per lo storage e il file system Amazon FSX per NetApp ONTAP, raggiungibile dai nodi di lavoro del cluster.

- Nodi di lavoro preparati per "NFS e/o iSCSI".

> Assicurati di seguire la procedura di preparazione del nodo richiesta per Amazon Linux e Ubuntu "Immagini Amazon Machine" (Amis) a seconda del tipo di AMI EKS.

Per altri requisiti di Astra Trident, vedere "qui".

Fasi

1. Implementare Astra Trident utilizzando uno dei ../trident-get-started/kubernetes-deploy.html[metodi di implementazione^].
2. Configurare Astra Trident come segue:
   a. Raccogliere il nome DNS LIF di gestione della SVM. Ad esempio, utilizzando l'interfaccia CLI AWS, individuare `DNSName` voce sotto `Endpoints` → `Management` dopo aver eseguito il seguente comando:

```
aws fsx describe-storage-virtual-machines --region <file system
region>
```

3. Creare e installare certificati per l'autenticazione. Se si utilizza un `ontap-san` back-end, vedere "qui". Se si utilizza un `ontap-nas` back-end, vedere "qui".

> È possibile accedere al file system (ad esempio per installare i certificati) utilizzando SSH da qualsiasi punto del file system. Utilizzare `fsxadmin` User (utente), la password configurata al momento della creazione del file system e il nome DNS di gestione da `aws fsx describe-file-systems`.

4. Creare un file backend utilizzando i certificati e il nome DNS della LIF di gestione, come mostrato nell'esempio seguente:

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "customBackendName",
  "managementLIF": "svm-XXXXXXXXXXXXXXXXX.fs-XXXXXXXXXXXXXXXXX.fsx.us-
east-2.aws.internal",
  "svm": "svm01",
  "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
  "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
  "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz",
 }
```

Per informazioni sulla creazione di backend, consulta i seguenti "Configurare un backend con i driver NAS ONTAP"
* "Configurare un backend con i driver SAN ONTAP"

Non specificare `dataLIF` per `ontap-san` e. `ontap-san-economy` Driver per consentire ad Astra Trident di utilizzare multipath.

Il `limitAggregateUsage` il parametro non funziona con `vsadmin` e. `fsxadmin` account utente. L'operazione di configurazione non riesce se si specifica questo parametro.

Dopo l'implementazione, eseguire la procedura per creare un "classe di storage, provisioning di un volume e montaggio del volume in un pod".

Trova ulteriori informazioni

- "Documentazione di Amazon FSX per NetApp ONTAP"
- "Post del blog su Amazon FSX per NetApp ONTAP"

Crea backend con kubectl

Un backend definisce la relazione tra Astra Trident e un sistema storage. Spiega ad Astra Trident come comunicare con quel sistema storage e come Astra Trident dovrebbe eseguire il provisioning dei volumi da esso. Dopo aver installato Astra Trident, il passo successivo è quello di creare un backend. Il `TridentBackendConfig` Custom Resource Definition (CRD) consente di creare e gestire backend Trident direttamente attraverso l'interfaccia Kubernetes. Per eseguire questa operazione, utilizzare `kubectl` O il tool CLI equivalente per la distribuzione Kubernetes.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`) È un CRD front-end, namespace, che consente di gestire i backend Astra Trident utilizzando `kubectl`. Kubernetes e gli amministratori dello storage possono ora creare e gestire i backend direttamente attraverso la CLI di Kubernetes senza richiedere un'utility a riga di comando dedicata (`tridentctl`).

Alla creazione di un `TridentBackendConfig` oggetto, si verifica quanto segue:

- Astra Trident crea automaticamente un backend in base alla configurazione che fornisci. Questo è rappresentato internamente come a. `TridentBackend` (`tbe`, `tridentbackend`) CR.
- Il `TridentBackendConfig` è vincolato in modo univoco a un `TridentBackend` Creato da Astra Trident.

Ciascuno `TridentBackendConfig` mantiene una mappatura uno a uno con un `TridentBackend`. Il primo è l'interfaccia fornita all'utente per progettare e configurare i backend; il secondo è il modo in cui Trident rappresenta l'oggetto backend effettivo.

`TridentBackend` I CRS vengono creati automaticamente da Astra Trident. Non è possibile modificarle. Se si desidera aggiornare i backend, modificare il `TridentBackendConfig` oggetto.

Vedere l'esempio seguente per il formato di `TridentBackendConfig` CR:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

È inoltre possibile esaminare gli esempi in "trident-installer" directory per configurazioni di esempio per la piattaforma/servizio di storage desiderato.

Il `spec` utilizza parametri di configurazione specifici per il back-end. In questo esempio, il backend utilizza `ontap-san` storage driver e utilizza i parametri di configurazione riportati in tabella. Per l'elenco delle opzioni di configurazione per il driver di storage desiderato, consultare "informazioni di configurazione back-end per il driver di storage".

Il `spec` la sezione include anche `credentials` e. `deletionPolicy` i campi, che sono stati introdotti di recente in `TridentBackendConfig` CR:

- `credentials`: Questo parametro è un campo obbligatorio e contiene le credenziali utilizzate per l'autenticazione con il sistema/servizio di storage. Questo è impostato su un Kubernetes Secret creato dall'utente. Le credenziali non possono essere passate in testo normale e si verificherà un errore.
- `deletionPolicy`: Questo campo definisce cosa deve accadere quando `TridentBackendConfig` viene cancellato. Può assumere uno dei due valori possibili:
  - `delete`: Questo comporta l'eliminazione di entrambi `TridentBackendConfig` CR e il backend associato. Questo è il valore predefinito.
  - `retain`: Quando un `TridentBackendConfig` La CR viene eliminata, la definizione di back-end rimane presente e può essere gestita con `tridentctl`. Impostazione del criterio di eliminazione su `retain` consente agli utenti di eseguire il downgrade a una release precedente (precedente alla 21.04) e conservare i backend creati. Il valore di questo campo può essere aggiornato dopo un `TridentBackendConfig` viene creato.

> Il nome di un backend viene impostato utilizzando `spec.backendName`. Se non specificato, il nome del backend viene impostato sul nome di `TridentBackendConfig` oggetto (metadata.name). Si consiglia di impostare esplicitamente i nomi backend utilizzando `spec.backendName`.

Backend creati con `tridentctl` non hanno un associato `TridentBackendConfig` oggetto. È possibile scegliere di gestire tali backend con `kubectl` creando un `TridentBackendConfig` CR. È necessario specificare parametri di configurazione identici (ad esempio `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`e così via). Astra Trident eseguirà automaticamente il binding del nuovo `TridentBackendConfig` con il backend preesistente.

Panoramica dei passaggi

Per creare un nuovo backend utilizzando `kubectl`, eseguire le seguenti operazioni:

1. Creare un "Kubernetes Secret". Il segreto contiene le credenziali che Astra Trident deve comunicare con il cluster/servizio di storage.
2. Creare un `TridentBackendConfig` oggetto. Contiene specifiche relative al cluster/servizio di storage e fa riferimento al segreto creato nel passaggio precedente.

Dopo aver creato un backend, è possibile osservarne lo stato utilizzando `kubectl get tbc <tbc-name> -n <trident-namespace>` e raccogliere ulteriori dettagli.

Fase 1: Creare un Kubernetes Secret

Creare un segreto contenente le credenziali di accesso per il backend. Si tratta di una caratteristica esclusiva di ogni piattaforma/servizio di storage. Ecco un esempio:

```
$ kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: t@Ax@7q(>
```

Questa tabella riassume i campi che devono essere inclusi nel Secret per ciascuna piattaforma di storage:

Descrizione dei campi segreti della piattaforma di storage	Segreto	Descrizione dei campi
Azure NetApp Files	ID cliente	L'ID client dalla registrazione di un'applicazione
Cloud Volumes Service per GCP	id_chiave_privata	ID della chiave privata. Parte della chiave API per l'account di servizio GCP con ruolo di amministratore CVS

Descrizione dei campi segreti della piattaforma di storage	Segreto	Descrizione dei campi
Cloud Volumes Service per GCP	private_key	Chiave privata. Parte della chiave API per l'account di servizio GCP con ruolo di amministratore CVS
Elemento (NetApp HCI/SolidFire)	Endpoint	MVIP per il cluster SolidFire con credenziali tenant
ONTAP	nome utente	Nome utente per la connessione al cluster/SVM. Utilizzato per l'autenticazione basata su credenziali
ONTAP	password	Password per la connessione al cluster/SVM. Utilizzato per l'autenticazione basata su credenziali
ONTAP	ClientPrivateKey	Valore codificato in base64 della chiave privata del client. Utilizzato per l'autenticazione basata su certificato
ONTAP	ChapNomeUtente	Nome utente inbound. Obbligatorio se useCHAP=true. Per `ontap-san` e. `ontap-san-economy`
ONTAP	ChapInitialatorSecret	Segreto iniziatore CHAP. Obbligatorio se useCHAP=true. Per `ontap-san` e. `ontap-san-economy`
ONTAP	ChapTargetNomeUtente	Nome utente di destinazione. Obbligatorio se useCHAP=true. Per `ontap-san` e. `ontap-san-economy`
ONTAP	ChapTargetInitialatorSecret	CHAP target Initiator secret. Obbligatorio se useCHAP=true. Per `ontap-san` e. `ontap-san-economy`

Il Segreto creato in questo passaggio verrà indicato in `spec.credentials` campo di `TridentBackendConfig` oggetto creato nel passaggio successivo.

Fase 2: Creare `TridentBackendConfig` CR

A questo punto, è possibile creare il `TridentBackendConfig` CR. In questo esempio, un backend che utilizza `ontap-san` il driver viene creato utilizzando `TridentBackendConfig` oggetto mostrato di seguito:

```
$ kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Fase 3: Verificare lo stato di `TridentBackendConfig` CR

Ora che è stato creato il `TridentBackendConfig` CR, è possibile verificare lo stato. Vedere il seguente esempio:

```
$ kubectl -n trident get tbc backend-tbc-ontap-san
NAME                     BACKEND NAME          BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san   ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success
```

Un backend è stato creato e associato a `TridentBackendConfig` CR.

La fase può assumere uno dei seguenti valori:

- `Bound`: Il `TridentBackendConfig` CR è associato a un backend e contiene tale backend `configRef` impostare su `TridentBackendConfig` L'uid di CR.
- `Unbound`: Rappresentato utilizzando `""`. Il `TridentBackendConfig` l'oggetto non è associato a un backend. Tutti creati di recente `TridentBackendConfig` I CRS sono in questa fase per impostazione predefinita. Una volta modificata la fase, non sarà più possibile tornare a Unbound.
- `Deleting`: Il `TridentBackendConfig` CR `deletionPolicy` è stato impostato per l'eliminazione. Quando il `TridentBackendConfig` La CR viene eliminata, passa allo stato di eliminazione.

- Se non sono presenti richieste di rimborso di volumi persistenti (PVC) sul back-end, eliminare il `TridentBackendConfig` In questo modo Astra Trident elimina il backend e il `TridentBackendConfig` CR.
- Se uno o più PVC sono presenti sul backend, passa a uno stato di eliminazione. Il `TridentBackendConfig` Successivamente, la CR entra anche nella fase di eliminazione. Il backend e. `TridentBackendConfig` Vengono eliminati solo dopo l'eliminazione di tutti i PVC.

- `Lost`: Il backend associato a `TridentBackendConfig` La CR è stata eliminata accidentalmente o deliberatamente e il `TridentBackendConfig` CR ha ancora un riferimento al backend cancellato. Il `TridentBackendConfig` La CR può comunque essere eliminata indipendentemente da `deletionPolicy` valore.
- `Unknown`: Astra Trident non è in grado di determinare lo stato o l'esistenza del backend associato a `TridentBackendConfig` CR. Ad esempio, se il server API non risponde o se `tridentbackends.trident.netapp.io` CRD mancante. Ciò potrebbe richiedere l'intervento dell'utente.

In questa fase, viene creato un backend. È possibile gestire anche diverse operazioni, ad esempio "aggiornamenti back-end ed eliminazioni back-end".

(Facoltativo) fase 4: Ulteriori informazioni

È possibile eseguire il seguente comando per ottenere ulteriori informazioni sul backend:

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

```
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success   ontap-san        delete
```

Inoltre, è possibile ottenere un dump YAML/JSON di `TridentBackendConfig`.

```
$ kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: "2021-04-21T20:45:11Z"
  finalizers:
  - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound
```

`backendInfo` contiene `backendName` e a. `backendUUID` del back-end creato in risposta a `TridentBackendConfig` CR. Il `lastOperationStatus` il campo rappresenta lo stato dell'ultima operazione di `TridentBackendConfig` CR, che può essere attivato dall'utente (ad esempio, l'utente ha modificato qualcosa in `spec`) O attivato da Astra Trident (ad esempio, durante il riavvio di Astra Trident). Può essere Success (riuscito) o Failed (non riuscito). `phase` rappresenta lo stato della relazione tra `TridentBackendConfig` CR e il back-end. Nell'esempio precedente, `phase` Ha il valore associato, il che significa che il `TridentBackendConfig` CR è associato al backend.

È possibile eseguire `kubectl -n trident describe tbc <tbc-cr-name>` per ottenere i dettagli dei registri degli eventi.

> Non è possibile aggiornare o eliminare un backend che contiene un associato `TridentBackendConfig` utilizzo di oggetti `tridentctl`. Comprendere le fasi necessarie per passare da un'operazione all'altra `tridentctl` e. `TridentBackendConfig`, "vedi qui".

Eseguire la gestione del back-end con kubectl

Scopri come eseguire operazioni di gestione back-end utilizzando `kubectl`.

Eliminare un backend

Eliminando un `TridentBackendConfig`, Si richiede ad Astra Trident di eliminare/conservare i backend (in base a. `deletionPolicy`). Per eliminare un backend, assicurarsi che `deletionPolicy` è impostato per eliminare. Per eliminare solo il `TridentBackendConfig`, assicurarsi che `deletionPolicy` è impostato su retain. In questo modo si garantisce che il backend sia ancora presente e che possa essere gestito tramite `tridentctl`.

Eseguire il seguente comando:

```
$ kubectl delete tbc <tbc-name> -n trident
```

Astra Trident non elimina i Kubernetes Secrets utilizzati da `TridentBackendConfig`. L'utente Kubernetes è responsabile della pulizia dei segreti. Prestare attenzione quando si eliminano i segreti. È necessario eliminare i segreti solo se non vengono utilizzati dai backend.

Visualizzare i backend esistenti

Eseguire il seguente comando:

```
$ kubectl get tbc -n trident
```

Puoi anche correre `tridentctl get backend -n trident` oppure `tridentctl get backend -o yaml -n trident` per ottenere un elenco di tutti i backend esistenti. Questo elenco includerà anche i backend creati con `tridentctl`.

Aggiornare un backend

Possono esserci diversi motivi per aggiornare un backend:

- Le credenziali del sistema storage sono state modificate. Per aggiornare le credenziali, il Kubernetes Secret utilizzato in `TridentBackendConfig` l'oggetto deve essere aggiornato. Astra Trident aggiornerà automaticamente il backend con le credenziali più recenti fornite. Eseguire il seguente comando per aggiornare Kubernetes Secret:

```
$ kubectl apply -f <updated-secret-file.yaml> -n trident
```

- È necessario aggiornare i parametri (ad esempio il nome della SVM ONTAP utilizzata). In questo caso, `TridentBackendConfig` Gli oggetti possono essere aggiornati direttamente tramite Kubernetes.

```
$ kubectl apply -f <updated-backend-file.yaml>
```

In alternativa, apportare modifiche all'esistente `TridentBackendConfig` CR eseguendo il seguente comando:

```
$ kubectl edit tbc <tbc-name> -n trident
```

Se un aggiornamento back-end non riesce, il back-end continua a rimanere nella sua ultima configurazione nota. È possibile visualizzare i log per determinare la causa eseguendo `kubectl get tbc <tbc-name> -o yaml -n trident` oppure `kubectl describe tbc <tbc-name> -n trident`.

Dopo aver identificato e corretto il problema con il file di configurazione, è possibile eseguire nuovamente il comando update.

Eseguire la gestione back-end con tridentctl

Scopri come eseguire operazioni di gestione back-end utilizzando `tridentctl`.

Creare un backend

Dopo aver creato un "file di configurazione back-end", eseguire il seguente comando:

```
$ tridentctl create backend -f <backend-file> -n trident
```

Se la creazione del back-end non riesce, si è verificato un errore nella configurazione del back-end. È possibile visualizzare i log per determinare la causa eseguendo il seguente comando:

```
$ tridentctl logs -n trident
```

Dopo aver identificato e corretto il problema con il file di configurazione, è possibile eseguire semplicemente `create` di nuovo comando.

Eliminare un backend

Per eliminare un backend da Astra Trident, procedere come segue:

1. Recuperare il nome del backend:

   ```
   $ tridentctl get backend -n trident
   ```

2. Eliminare il backend:

   ```
   $ tridentctl delete backend <backend-name> -n trident
   ```

> Se Astra Trident ha eseguito il provisioning di volumi e snapshot da questo backend ancora esistenti, l'eliminazione del backend impedisce il provisioning di nuovi volumi da parte dell'IT. Il backend continuerà a esistere in uno stato di eliminazione e Trident continuerà a gestire tali volumi e snapshot fino a quando non verranno eliminati.

Visualizzare i backend esistenti

Per visualizzare i backend di cui Trident è a conoscenza, procedere come segue:

- Per ottenere un riepilogo, eseguire il seguente comando:

```
$ tridentctl get backend -n trident
```

- Per ottenere tutti i dettagli, eseguire il seguente comando:

```
$ tridentctl get backend -o json -n trident
```

Aggiornare un backend

Dopo aver creato un nuovo file di configurazione back-end, eseguire il seguente comando:

```
$ tridentctl update backend <backend-name> -f <backend-file> -n trident
```

Se l'aggiornamento del back-end non riesce, si è verificato un errore nella configurazione del back-end o si è tentato di eseguire un aggiornamento non valido. È possibile visualizzare i log per determinare la causa eseguendo il seguente comando:

```
$ tridentctl logs -n trident
```

Dopo aver identificato e corretto il problema con il file di configurazione, è possibile eseguire semplicemente `update` di nuovo comando.

Identificare le classi di storage che utilizzano un backend

Questo è un esempio del tipo di domande a cui puoi rispondere con il JSON che `tridentctl` output per oggetti backend. Viene utilizzato il `jq` che è necessario installare.

```
$ tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

Questo vale anche per i backend creati con `TridentBackendConfig`.

Passare da un'opzione di gestione back-end all'altra

Scopri i diversi modi di gestire i backend in Astra Trident. Con l'introduzione di `TridentBackendConfig`, gli amministratori dispongono ora di due metodi unici per gestire i back-end. Questo pone le seguenti domande:

- È possibile creare backend utilizzando `tridentctl` essere gestito con `TridentBackendConfig`?
- È possibile creare backend utilizzando `TridentBackendConfig` essere gestito con `tridentctl`?

Gestire `tridentctl` backend con `TridentBackendConfig`

In questa sezione vengono descritte le procedure necessarie per gestire i backend creati con `tridentctl` Direttamente attraverso l'interfaccia Kubernetes creando `TridentBackendConfig` oggetti.

Questo si applica ai seguenti scenari:

- Backend preesistenti, che non dispongono di `TridentBackendConfig` perché sono stati creati con `tridentctl`.
- Nuovi backend creati con `tridentctl`, mentre altri `TridentBackendConfig` esistono oggetti.

In entrambi gli scenari, i backend continueranno a essere presenti, con Astra Trident che pianifica i volumi e li gestisce. Gli amministratori possono scegliere tra due opzioni:

- Continuare a utilizzare `tridentctl` per gestire i back-end creati utilizzando l'it.
- Collegare i backend creati con `tridentctl` a un nuovo `TridentBackendConfig` oggetto. In questo modo, i backend verranno gestiti utilizzando `kubectl` e non `tridentctl`.

Per gestire un backend preesistente utilizzando `kubectl`, sarà necessario creare un `TridentBackendConfig` che si collega al back-end esistente. Ecco una panoramica sul funzionamento di questo sistema:

1. Crea un Kubernetes Secret. Il segreto contiene le credenziali che Astra Trident deve comunicare con il cluster/servizio di storage.
2. Creare un `TridentBackendConfig` oggetto. Contiene specifiche relative al cluster/servizio di storage e fa riferimento al segreto creato nel passaggio precedente. È necessario specificare parametri di configurazione identici (ad esempio `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`e così via). `spec.backendName` deve essere impostato sul nome del backend esistente.

Fase 0: Identificare il backend

Per creare un `TridentBackendConfig` che si collega a un backend esistente, sarà necessario ottenere la configurazione del backend. In questo esempio, supponiamo che sia stato creato un backend utilizzando la seguente definizione JSON:

```
$ tridentctl get backend ontap-nas-backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                  UUID
| STATE  | VOLUMES |
```

```
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+

$ cat ontap-nas-backend.json

{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.10.10.1",
    "dataLIF": "10.10.10.2",
    "backendName": "ontap-nas-backend",
    "svm": "trident_svm",
    "username": "cluster-admin",
    "password": "admin-password",

    "defaults": {
        "spaceReserve": "none",
        "encryption": "false"
    },
    "labels":{"store":"nas_store"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"app":"msoffice", "cost":"100"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"app":"mysqldb", "cost":"25"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
        }
    ]
}
```

Fase 1: Creare un Kubernetes Secret

Creare un Segreto contenente le credenziali per il backend, come illustrato in questo esempio:

```
$ cat tbc-ontap-nas-backend-secret.yaml

apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  passWord: admin-password

$ kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

Fase 2: Creare un `TridentBackendConfig` CR

Il passaggio successivo consiste nella creazione di un `TridentBackendConfig` CR che si associerà automaticamente al preesistente `ontap-nas-backend` (come in questo esempio). Assicurarsi che siano soddisfatti i seguenti requisiti:

- Lo stesso nome backend viene definito in `spec.backendName`.
- I parametri di configurazione sono identici al backend originale.
- I pool di storage virtuali (se presenti) devono mantenere lo stesso ordine del backend originale.
- Le credenziali vengono fornite attraverso un Kubernetes Secret e non in testo normale.

In questo caso, il `TridentBackendConfig` avrà un aspetto simile al seguente:

```
$ cat backend-tbc-ontap-nas.yaml
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
      app: msoffice
      cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
      app: mysqldb
      cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'

$ kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

Fase 3: Verificare lo stato di `TridentBackendConfig` CR

Dopo il `TridentBackendConfig` è stato creato, la sua fase deve essere `Bound`. Deve inoltre riflettere lo stesso nome e UUID del backend esistente.

```
$ kubectl -n trident get tbc tbc-ontap-nas-backend -n trident
NAME                    BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
$ tridentctl get backend -n trident
+----------------------+----------------
+--------------------------------------+--------+---------+
|          NAME        | STORAGE DRIVER |                  UUID
| STATE  | VOLUMES |
+----------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+----------------------+----------------
+--------------------------------------+--------+---------+
```

Il back-end verrà ora completamente gestito utilizzando `tbc-ontap-nas-backend` `TridentBackendConfig` oggetto.

Gestire `TridentBackendConfig` backend con `tridentctl`

```
`tridentctl` può essere utilizzato per elencare i backend creati con
`TridentBackendConfig`. Inoltre, gli amministratori possono anche
scegliere di gestire completamente tali backend attraverso `tridentctl`
eliminando `TridentBackendConfig` e assicurandosi `spec.deletionPolicy` è
impostato su `retain`.
```

Fase 0: Identificare il backend

Ad esempio, supponiamo che il seguente backend sia stato creato utilizzando `TridentBackendConfig`:

```
$ kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

$ tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

Dall'output, si vede che `TridentBackendConfig` È stato creato correttamente ed è associato a un backend [osservare l'UUID del backend].

Fase 1: Confermare `deletionPolicy` è impostato su `retain`

Diamo un'occhiata al valore di `deletionPolicy`. Questo valore deve essere impostato su `retain`. In questo modo si garantisce che quando si verifica un `TridentBackendConfig` La CR viene eliminata, la definizione di back-end rimane presente e può essere gestita con `tridentctl`.

```
$ kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
$ kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
$ kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

Non passare alla fase successiva a meno che `deletionPolicy` è impostato su `retain`.

Fase 2: Eliminare `TridentBackendConfig` CR

Il passaggio finale consiste nell'eliminare `TridentBackendConfig` CR. Dopo la conferma di `deletionPolicy` è impostato su `retain`, è possibile procedere con l'eliminazione:

```
$ kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

$ tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

Al momento dell'eliminazione di `TridentBackendConfig` Astra Trident lo rimuove senza eliminare il backend stesso.

Gestire le classi di storage

Informazioni sulla creazione di una classe di storage, sull'eliminazione di una classe di storage e sulla visualizzazione delle classi di storage esistenti.

Progettare una classe di storage

Vedere "qui" per ulteriori informazioni sulle classi di storage e su come configurarle.

Creare una classe di storage

Dopo aver creato un file di classe storage, eseguire il seguente comando:

```
kubectl create -f <storage-class-file>
```

`<storage-class-file>` deve essere sostituito con il nome del file della classe di storage.

Eliminare una classe di storage

Per eliminare una classe di storage da Kubernetes, eseguire il seguente comando:

```
kubectl delete storageclass <storage-class>
```

`<storage-class>` deve essere sostituito con la classe di storage.

Tutti i volumi persistenti creati attraverso questa classe di storage resteranno inalterati e Astra Trident continuerà a gestirli.

> Astra Trident impone un vuoto `fsType` per i volumi creati. Per i backend iSCSI, si consiglia di applicare `parameters.fsType` In StorageClass. È necessario eliminare e ricreare StorageClasses con `parameters.fsType` specificato.

Visualizzare le classi di storage esistenti

- Per visualizzare le classi di storage Kubernetes esistenti, eseguire il seguente comando:

```
kubectl get storageclass
```

- Per visualizzare i dettagli della classe storage Kubernetes, eseguire il seguente comando:

```
kubectl get storageclass <storage-class> -o json
```

- Per visualizzare le classi di storage sincronizzate di Astra Trident, eseguire il seguente comando:

```
tridentctl get storageclass
```

- Per visualizzare i dettagli della classe di storage sincronizzata di Astra Trident, eseguire il seguente comando:

```
tridentctl get storageclass <storage-class> -o json
```

Impostare una classe di storage predefinita

Kubernetes 1.6 ha aggiunto la possibilità di impostare una classe di storage predefinita. Si tratta della classe di storage che verrà utilizzata per eseguire il provisioning di un volume persistente se un utente non ne specifica uno in un PVC (Persistent Volume Claim).

- Definire una classe di storage predefinita impostando l'annotazione `storageclass.kubernetes.io/is-default-class` a true nella definizione della classe di storage. In base alla specifica, qualsiasi altro valore o assenza di annotazione viene interpretato come falso.
- È possibile configurare una classe di storage esistente come classe di storage predefinita utilizzando il seguente comando:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- Allo stesso modo, è possibile rimuovere l'annotazione predefinita della classe di storage utilizzando il seguente comando:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Nel bundle del programma di installazione di Trident sono presenti anche alcuni esempi che includono questa annotazione.

> Nel cluster dovrebbe essere presente una sola classe di storage predefinita per volta. Kubernetes non impedisce tecnicamente di averne più di una, ma si comporta come se non ci fosse alcuna classe di storage predefinita.

Identificare il backend per una classe di storage

Questo è un esempio del tipo di domande a cui puoi rispondere con il JSON che `tridentctl` Output per gli oggetti backend Astra Trident. Viene utilizzato il `jq` che potrebbe essere necessario installare per prima.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

Eseguire operazioni sui volumi

Scopri le funzionalità offerte da Astra Trident per la gestione dei volumi.

- "Utilizzare la topologia CSI"
- "Lavorare con le istantanee"
- "Espandere i volumi"
- "Importa volumi"

Utilizzare la topologia CSI

Astra Trident può creare e collegare in modo selettivo volumi ai nodi presenti in un cluster Kubernetes utilizzando "Funzionalità topologia CSI". Utilizzando la funzionalità topologia CSI, l'accesso ai volumi può essere limitato a un sottoinsieme di nodi, in base alle aree geografiche e alle zone di disponibilità. I provider di cloud oggi consentono agli amministratori di Kubernetes di generare nodi basati su zone. I nodi possono essere collocati in diverse zone di disponibilità all'interno di una regione o in diverse regioni. Per facilitare il provisioning dei volumi per i carichi di lavoro in un'architettura multi-zona, Astra Trident utilizza la topologia CSI.

> Scopri di più sulla funzionalità topologia CSI "qui".

Kubernetes offre due esclusive modalità di binding del volume:

- Con `VolumeBindingMode` impostare su `Immediate`, Astra Trident crea il volume senza alcuna consapevolezza della topologia. Il binding dei volumi e il provisioning dinamico vengono gestiti quando viene creato il PVC. Questa è l'impostazione predefinita `VolumeBindingMode` ed è adatto per i cluster che non applicano vincoli di topologia. I volumi persistenti vengono creati senza alcuna dipendenza dai requisiti di pianificazione del pod richiedente.
- Con `VolumeBindingMode` impostare su `WaitForFirstConsumer`, La creazione e il binding di un volume persistente per un PVC viene ritardata fino a quando un pod che utilizza il PVC viene pianificato e creato. In questo modo, i volumi vengono creati per soddisfare i vincoli di pianificazione imposti dai requisiti di topologia.

> Il `WaitForFirstConsumer` la modalità di binding non richiede etichette di topologia. Questo può essere utilizzato indipendentemente dalla funzionalità topologia CSI.

Di cosa hai bisogno

Per utilizzare la topologia CSI, è necessario disporre di quanto segue:

- Un cluster Kubernetes con 1.17 o versione successiva.

```
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- I nodi nel cluster devono essere dotati di etichette che introducano la consapevolezza della topologia (`topology.kubernetes.io/region` e. `topology.kubernetes.io/zone`). Queste etichette **devono essere presenti sui nodi del cluster** prima dell'installazione di Astra Trident affinché Astra Trident sia consapevole della topologia.

```
$ kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

Fase 1: Creazione di un backend compatibile con la topologia

I backend di storage Astra Trident possono essere progettati per eseguire il provisioning selettivo dei volumi in base alle zone di disponibilità. Ogni backend può portare un optional `supportedTopologies` blocco che rappresenta un elenco di zone e regioni che devono essere supportate. Per StorageClasses che utilizzano tale backend, un volume viene creato solo se richiesto da un'applicazione pianificata in una regione/zona supportata.

Di seguito viene riportato un esempio di definizione di backend:

```
{
 "version": 1,
 "storageDriverName": "ontap-san",
 "backendName": "san-backend-us-east1",
 "managementLIF": "192.168.27.5",
 "svm": "iscsi_svm",
 "username": "admin",
 "password": "xxxxxxxxxxxx",
 "supportedTopologies": [
{"topology.kubernetes.io/region": "us-east1",
"topology.kubernetes.io/zone": "us-east1-a"},
{"topology.kubernetes.io/region": "us-east1",
"topology.kubernetes.io/zone": "us-east1-b"}
]
}
```

`supportedTopologies` viene utilizzato per fornire un elenco di regioni e zone per backend. Queste regioni e zone rappresentano l'elenco dei valori consentiti che possono essere forniti in una StorageClass. Per StorageClasses che contengono un sottoinsieme delle regioni e delle zone fornite in un backend, Astra Trident creerà un volume sul backend.

È possibile definire `supportedTopologies` anche per pool di storage. Vedere il seguente esempio:

```
{"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "nas-backend-us-central1",
"managementLIF": "172.16.238.5",
"svm": "nfs_svm",
"username": "admin",
"password": "Netapp123",
"supportedTopologies": [
      {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-a"},
      {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-b"}
    ]
"storage": [
   {
       "labels": {"workload":"production"},
        "region": "Iowa-DC",
        "zone": "Iowa-DC-A",
        "supportedTopologies": [
            {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-a"}
        ]
    },
    {
        "labels": {"workload":"dev"},
         "region": "Iowa-DC",
         "zone": "Iowa-DC-B",
         "supportedTopologies": [
             {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-b"}
         ]
     }
]
}
```

In questo esempio, il `region` e. `zone` le etichette indicano la posizione del pool di storage. `topology.kubernetes.io/region` e. `topology.kubernetes.io/zone` stabilire da dove possono essere consumati i pool di storage.

Fase 2: Definire StorageClasses che siano compatibili con la topologia

In base alle etichette della topologia fornite ai nodi del cluster, è possibile definire StorageClasses in modo da contenere informazioni sulla topologia. In questo modo verranno determinati i pool di storage che fungono da candidati per le richieste PVC effettuate e il sottoinsieme di nodi che possono utilizzare i volumi forniti da Trident.

Vedere il seguente esempio:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
- key: topology.kubernetes.io/zone
  values:
  - us-east1-a
  - us-east1-b
- key: topology.kubernetes.io/region
  values:
  - us-east1
parameters:
  fsType: "ext4"
```

Nella definizione di StorageClass sopra riportata, `volumeBindingMode` è impostato su `WaitForFirstConsumer`. I PVC richiesti con questa classe di storage non verranno utilizzati fino a quando non saranno referenziati in un pod. Inoltre, `allowedTopologies` fornisce le zone e la regione da utilizzare. Il `netapp-san-us-east1` StorageClass crea PVC su `san-backend-us-east1` backend definito sopra.

Fase 3: Creare e utilizzare un PVC

Con StorageClass creato e mappato a un backend, è ora possibile creare PVC.

Vedere l'esempio `spec` sotto:

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

La creazione di un PVC utilizzando questo manifesto comporta quanto segue:

```
$ kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
$ kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
$ kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

Affinché Trident crei un volume e lo leghi al PVC, utilizza il PVC in un pod. Vedere il seguente esempio:

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

Questo podSpec indica a Kubernetes di pianificare il pod sui nodi presenti in `us-east1` e scegliere tra i nodi presenti in `us-east1-a` oppure `us-east1-b` zone.

Vedere il seguente output:

```
$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
$ kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

Aggiorna i back-end da includere `supportedTopologies`

I backend preesistenti possono essere aggiornati per includere un elenco di `supportedTopologies` utilizzo di `tridentctl backend update`. Ciò non influisce sui volumi già sottoposti a provisioning e verrà utilizzato solo per i PVC successivi.

Trova ulteriori informazioni

- "Gestire le risorse per i container"
- "NodeSelector"
- "Affinità e anti-affinità"
- "Contamini e pedaggi"

Lavorare con le istantanee

A partire dalla versione 20.01 di Astra Trident, è possibile creare snapshot di PVS nel livello Kubernetes. È possibile utilizzare queste snapshot per mantenere copie point-in-time dei volumi creati da Astra Trident e pianificare la creazione di volumi aggiuntivi (cloni). Lo snapshot del volume è supportato da `ontap-nas`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, e. `azure-netapp-files` driver.

> Questa funzione è disponibile da Kubernetes 1.17 (beta) ed è GA da 1.20. Per informazioni sulle modifiche apportate al passaggio dalla versione beta a quella GA, vedere "il blog di release". Con la laurea in GA, il `v1` La versione API è stata introdotta ed è compatibile con le versioni precedenti `v1beta1` snapshot.

Di cosa hai bisogno

- La creazione di snapshot dei volumi richiede la creazione di un controller di snapshot esterno e di alcune definizioni di risorse personalizzate (CRD). Questa è la responsabilità del Kubernetes orchestrator in uso (ad esempio: Kubeadm, GKE, OpenShift).

È possibile creare uno snapshot-controller esterno e uno snapshot CRD come segue:

1. Creare CRD snapshot di volume:

```
$ cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Creare lo snapshot-controller nello spazio dei nomi desiderato. Modificare i manifesti YAML riportati di seguito per modificare lo spazio dei nomi.

> Non creare uno snapshot-controller se si configurano snapshot di volumi on-demand in un ambiente GKE. GKE utilizza un controller di snapshot integrato e nascosto.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-3.0/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-3.0/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

> CSI Snapshotter offre un "convalida di webhook" per aiutare gli utenti a convalidare le istantanee v1beta1 esistenti e confermare che si tratta di oggetti di risorse validi. Il webhook validante etichetta automaticamente gli oggetti snapshot non validi e impedisce la creazione di oggetti non validi futuri. Il webhook di convalida viene implementato da Kubernetes orchestrator. Consultare le istruzioni per implementare manualmente il webhook di convalida "qui". Trova esempi di manifesti di snapshot non validi "qui".

Nell'esempio riportato di seguito vengono illustrati i costrutti necessari per l'utilizzo delle snapshot e viene illustrato come creare e utilizzare le istantanee.

Fase 1: Impostare un `VolumeSnapshotClass`

Prima di creare un'istantanea del volume, impostare un collegamento:../trident-reference/objects.html[`VolumeSnapshotClass`^].

```
$ cat snap-sc.yaml
#Use apiVersion v1 for Kubernetes 1.20 and above. For Kubernetes 1.17 -
1.19, use apiVersion v1beta1.
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

Il `driver` Indica il driver CSI di Astra Trident. `deletionPolicy` può essere `Delete` oppure `Retain`. Quando è impostato su `Retain`, lo snapshot fisico sottostante sul cluster di storage viene conservato anche quando `VolumeSnapshot` oggetto eliminato.

Fase 2: Creare un'istantanea di un PVC esistente

```
$ cat snap.yaml
#Use apiVersion v1 for Kubernetes 1.20 and above. For Kubernetes 1.17 -
1.19, use apiVersion v1beta1.
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

Lo snapshot è in fase di creazione per un PVC denominato `pvc1`e il nome dello snapshot è impostato su `pvc1-snap`.

```
$ kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

$ kubectl get volumesnapshots
NAME                   AGE
pvc1-snap              50s
```

Questo ha creato un `VolumeSnapshot` oggetto. Un'istantanea VolumeSnapshot è analoga a un PVC ed è associata a un `VolumeSnapshotContent` oggetto che rappresenta lo snapshot effettivo.

È possibile identificare `VolumeSnapshotContent` oggetto per `pvc1-snap` VolumeSnapshot descrivendolo.

```
$ kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
.
.
.
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
.
.
```

Il `Snapshot Content Name` Identifica l'oggetto VolumeSnapshotContent che fornisce questa snapshot. Il `Ready To Use` Il parametro indica che l'istantanea può essere utilizzata per creare un nuovo PVC.

Fase 3: Creazione di PVC da VolumeSnapshots

Vedere l'esempio seguente per creare un PVC utilizzando uno snapshot:

```
$ cat pvc-from-snap.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

`dataSource` Mostra che il PVC deve essere creato utilizzando un VolumeSnapshot denominato `pvc1-snap`

come origine dei dati. Questo indica ad Astra Trident di creare un PVC dall'istantanea. Una volta creato, il PVC può essere collegato a un pod e utilizzato come qualsiasi altro PVC.

> Quando si elimina un volume persistente con snapshot associate, il volume Trident corrispondente viene aggiornato a uno stato di eliminazione. Per eliminare il volume Astra Trident, è necessario rimuovere le snapshot del volume.

Trova ulteriori informazioni

- "Snapshot dei volumi"
- `VolumeSnapshotClass`

Espandere i volumi

Astra Trident offre agli utenti Kubernetes la possibilità di espandere i propri volumi dopo la loro creazione. Informazioni sulle configurazioni richieste per espandere i volumi iSCSI e NFS.

Espandere un volume iSCSI

È possibile espandere un volume persistente iSCSI (PV) utilizzando il provisioning CSI.

> L'espansione del volume iSCSI è supportata da `ontap-san`, `ontap-san-economy`, `solidfire-san` Driver e richiede Kubernetes 1.16 e versioni successive.

Panoramica

L'espansione di un PV iSCSI include i seguenti passaggi:

- Modifica della definizione di StorageClass per impostare `allowVolumeExpansion` campo a. `true`.
- Modifica della definizione PVC e aggiornamento di `spec.resources.requests.storage` per riflettere le nuove dimensioni desiderate, che devono essere superiori alle dimensioni originali.
- Il collegamento del PV deve essere collegato a un pod per poter essere ridimensionato. Esistono due scenari quando si ridimensiona un PV iSCSI:
  - Se il PV è collegato a un pod, Astra Trident espande il volume sul backend dello storage, esegue di nuovo la scansione del dispositivo e ridimensiona il file system.
  - Quando si tenta di ridimensionare un PV non collegato, Astra Trident espande il volume sul backend dello storage. Dopo aver associato il PVC a un pod, Trident esegue nuovamente la scansione del dispositivo e ridimensiona il file system. Kubernetes aggiorna quindi le dimensioni del PVC dopo il completamento dell'operazione di espansione.

L'esempio seguente mostra come funziona l'espansione del PVS iSCSI.

Fase 1: Configurare StorageClass per supportare l'espansione dei volumi

```
$ cat storageclass-ontapsan.yaml
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Per un StorageClass già esistente, modificarlo per includere `allowVolumeExpansion` parametro.

Fase 2: Creare un PVC con la StorageClass creata

```
$ cat pvc-ontapsan.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Astra Trident crea un volume persistente (PV) e lo associa a questo PVC (Persistent Volume Claim).

```
$ kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               10s
```

Fase 3: Definire un pod che colleghi il PVC

In questo esempio, viene creato un pod che utilizza `san-pvc`.

```
$  kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
centos-pod   1/1     Running   0          65s

$  kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    centos-pod
```

Fase 4: Espandere il PV

Per ridimensionare il PV creato da 1 Gi a 2 Gi, modificare la definizione PVC e aggiornare `spec.resources.requests.storage` A 2 Gi.

```
$ kubectl edit pvc san-pvc
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 ...
```

Fase 5: Convalidare l’espansione

È possibile verificare che l’espansione funzioni correttamente controllando le dimensioni del volume PVC, PV e Astra Trident:

```
$ kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
$ tridentctl get volumes -n trident
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

Espandere un volume NFS

Astra Trident supporta l'espansione dei volumi per NFS PVS su cui è stato eseguito il provisioning `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `gcp-cvs`, e. `azure-netapp-files` back-end.

Fase 1: Configurare StorageClass per supportare l'espansione dei volumi

Per ridimensionare un PV NFS, l'amministratore deve prima configurare la classe di storage per consentire l'espansione del volume impostando `allowVolumeExpansion` campo a. `true`:

```
$ cat storageclass-ontapnas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

Se è già stata creata una classe di storage senza questa opzione, è possibile modificare semplicemente la classe di storage esistente utilizzando `kubectl edit storageclass` per consentire l'espansione del volume.

Fase 2: Creare un PVC con la StorageClass creata

```
$ cat pvc-ontapnas.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Astra Trident deve creare un PV NFS 20MiB per questo PVC:

```
$ kubectl get pvc
NAME           STATUS   VOLUME
CAPACITY     ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

$ kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

Fase 3: Espandere il PV

Per ridimensionare il PV 20MiB appena creato in 1GiB, modificare il PVC e impostare `spec.resources.requests.storage` A 1 GB:

```
$ kubectl edit pvc ontapnas20mb
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
...
```

Fase 4: Convalidare l’espansione

È possibile verificare che il ridimensionamento funzioni correttamente controllando le dimensioni del volume PVC, PV e Astra Trident:

```
$ kubectl get pvc ontapnas20mb
NAME           STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi
RWO            ontapnas        4m44s

$ kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

$ tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n
trident
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

Importa volumi

È possibile importare volumi di storage esistenti come PV Kubernetes utilizzando `tridentctl import`.

Driver che supportano l'importazione di volumi

Questa tabella illustra i driver che supportano l'importazione di volumi e la release in cui sono stati introdotti.

Driver	Rilasciare
`ontap-nas`	19.04
`ontap-nas-flexgroup`	19.04
`solidfire-san`	19.04
`azure-netapp-files`	19.04

Driver	Rilasciare
gcp-cvs	19.04
ontap-san	19.04

Perché è necessario importare i volumi?

Esistono diversi casi di utilizzo per l'importazione di un volume in Trident:

- La creazione di un'applicazione e il riutilizzo del set di dati esistente
- Utilizzo di un clone di un set di dati per un'applicazione temporanea
- Ricostruzione di un cluster Kubernetes guasto
- Migrazione dei dati delle applicazioni durante il disaster recovery

Come funziona l'importazione?

Il file PVC (Persistent Volume Claim) viene utilizzato dal processo di importazione del volume per creare il PVC. Come minimo, il file PVC deve includere i campi name, namespace, accessModes e storageClassName, come illustrato nell'esempio seguente.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```

Il `tridentctl` client viene utilizzato per importare un volume di storage esistente. Trident importa il volume persistendo i metadati del volume e creando PVC e PV.

```
$ tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-
file>
```

Per importare un volume di storage, specificare il nome del backend Astra Trident contenente il volume, nonché il nome che identifica in modo univoco il volume nello storage (ad esempio: ONTAP FlexVol, Volume elemento, percorso del volume CVS). Il volume di storage deve consentire l'accesso in lettura/scrittura ed essere accessibile dal backend Astra Trident specificato. Il `-f` L'argomento string è obbligatorio e specifica il percorso del file YAML o JSON PVC.

Quando Astra Trident riceve la richiesta del volume di importazione, le dimensioni del volume esistente vengono determinate e impostate nel PVC. Una volta importato il volume dal driver di storage, il PV viene creato con un ClaimRef sul PVC. La policy di recupero viene inizialmente impostata su `retain` Nel PV. Dopo

che Kubernetes ha eseguito il binding con PVC e PV, la policy di recupero viene aggiornata in modo da corrispondere alla policy di recupero della classe di storage. Se il criterio di recupero della classe di storage è `delete`, Il volume di storage viene cancellato quando il PV viene cancellato.

Quando viene importato un volume con `--no-manage` Argomento: Trident non esegue operazioni aggiuntive sul PVC o sul PV per il ciclo di vita degli oggetti. Perché Trident ignora gli eventi PV e PVC per `--no-manage` Oggetti, il volume di storage non viene cancellato quando il PV viene cancellato. Vengono ignorate anche altre operazioni, come il clone del volume e il ridimensionamento del volume. Questa opzione è utile se si desidera utilizzare Kubernetes per carichi di lavoro containerizzati, ma altrimenti si desidera gestire il ciclo di vita del volume di storage al di fuori di Kubernetes.

Al PVC e al PV viene aggiunta un'annotazione che serve a doppio scopo per indicare che il volume è stato importato e se il PVC e il PV sono gestiti. Questa annotazione non deve essere modificata o rimossa.

Trident 19.07 e versioni successive gestiscono il collegamento di PVS e montano il volume come parte dell'importazione. Per le importazioni che utilizzano versioni precedenti di Astra Trident, non verranno eseguite operazioni nel percorso dei dati e l'importazione del volume non verificherà se il volume può essere montato. Se si commette un errore con l'importazione del volume (ad esempio, StorageClass non è corretta), è possibile ripristinare modificando la policy di recupero sul PV in `retain`, Eliminando PVC e PV e riprovando il comando di importazione del volume.

`ontap-nas` e. `ontap-nas-flexgroup` importazioni

Ogni volume creato con `ontap-nas` Driver è un FlexVol sul cluster ONTAP. Importazione di FlexVol con `ontap-nas` il driver funziona allo stesso modo. Un FlexVol già presente in un cluster ONTAP può essere importato come `ontap-nas` PVC. Allo stesso modo, è possibile importare i volumi FlexGroup come `ontap-nas-flexgroup` PVC.

> Un volume ONTAP deve essere di tipo rw per essere importato da Trident. Se un volume è di tipo dp, si tratta di un volume di destinazione SnapMirror; è necessario interrompere la relazione di mirroring prima di importare il volume in Trident.

> Il `ontap-nas` il driver non può importare e gestire qtree. Il `ontap-nas` e. `ontap-nas-flexgroup` i driver non consentono nomi di volumi duplicati.

Ad esempio, per importare un volume denominato `managed_volume` su un backend denominato `ontap_nas`, utilizzare il seguente comando:

```
$ tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

Per importare un volume denominato `unmanaged_volume` (su `ontap_nas backend`), che Trident non gestirà, utilizzare il seguente comando:

```
$ tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-file>
--no-manage

+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

Quando si utilizza `--no-manage` Argomento: Trident non rinomina il volume né convalida se il volume è stato montato. L'operazione di importazione del volume non riesce se il volume non è stato montato manualmente.

> È stato risolto un bug esistente in precedenza relativo all'importazione di volumi con UnixPermissions personalizzati. È possibile specificare unixPermissions nella definizione PVC o nella configurazione backend e richiedere ad Astra Trident di importare il volume di conseguenza.

`ontap-san` importa

Astra Trident può anche importare SAN FlexVol ONTAP che contengono una singola LUN. Ciò è coerente con `ontap-san` Driver, che crea un FlexVol per ogni PVC e un LUN all'interno di FlexVol. È possibile utilizzare `tridentctl import` comando nello stesso modo degli altri casi:

- Includere il nome di `ontap-san` back-end.

- Specificare il nome del FlexVol da importare. Tenere presente che questo FlexVol contiene un solo LUN che deve essere importato.
- Fornire il percorso della definizione PVC che deve essere utilizzata con `-f` allarme.
- Scegli tra gestire il PVC o non gestirlo. Per impostazione predefinita, Trident gestirà il PVC e rinominerà il FlexVol e il LUN sul backend. Per importare come volume non gestito, passare a. `--no-manage` allarme.

> Quando si importa un non gestito `ontap-san` Assicurarsi che il LUN nel FlexVol sia denominato `lun0` ed è mappato ad un igroup con gli iniziatori desiderati. Astra Trident gestisce automaticamente questa operazione per un'importazione gestita.

Astra Trident importa il FlexVol e lo associa alla definizione del PVC. Astra Trident rinomina anche FlexVol in `pvc-<uuid>` E il LUN all'interno di FlexVol a. `lun0`.

> Si consiglia di importare volumi che non dispongono di connessioni attive. Se si desidera importare un volume utilizzato attivamente, clonare prima il volume, quindi eseguire l'importazione.

Esempio

Per importare `ontap-san-managed` FlexVol presente su `ontap_san_default` eseguire il backend `tridentctl import` comando come:

```
$ tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
```

> Per poter essere importato da Astra Trident, un volume ONTAP deve essere di tipo rw. Se un volume è di tipo dp, si tratta di un volume di destinazione SnapMirror; è necessario interrompere la relazione di mirroring prima di importare il volume in Astra Trident.

`element` importa

Con Trident è possibile importare il software NetApp Element/volumi NetApp HCI nel cluster Kubernetes. È necessario il nome del backend Astra Trident e il nome univoco del volume e del file PVC come argomenti per `tridentctl import` comando.

```
$ tridentctl import volume element_default element-managed -f pvc-basic-
import.yaml -n trident -d

+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
```

Il driver Element supporta nomi di volumi duplicati. Se sono presenti nomi di volumi duplicati, il processo di importazione dei volumi di Trident restituisce un errore. Come soluzione alternativa, clonare il volume e fornire un nome di volume univoco. Quindi importare il volume clonato.

`gcp-cvs` **importa**

Per importare un volume supportato da NetApp Cloud Volumes Service in GCP, identificare il volume in base al percorso del volume anziché al nome.

Per importare un `gcp-cvs` volume sul backend chiamato `gcpcvs_YEppr` con il percorso del volume di `adroit-jolly-swift`, utilizzare il seguente comando:

```
$ tridentctl import volume gcpcvs_YEppr adroit-jolly-swift -f <path-to-
pvc-file> -n trident

+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | 93 GiB | gcp-storage   | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true    |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
```

Il percorso del volume è la parte del percorso di esportazione del volume dopo :/. Ad esempio, se il percorso di esportazione è `10.0.0.1:/adroit-jolly-swift`, il percorso del volume è `adroit-jolly-swift`.

`azure-netapp-files` **importa**

Per importare un `azure-netapp-files` volume sul backend chiamato `azurenetappfiles_40517` con il percorso del volume `importvol1`, eseguire il seguente comando:

```
$ tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage   |
file     | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

> Il percorso del volume per il volume ANF è presente nel percorso di montaggio dopo :/. Ad esempio, se il percorso di montaggio è `10.0.0.2:/importvol1`, il percorso del volume è `importvol1`.

Preparare il nodo di lavoro

Tutti i nodi di lavoro nel cluster Kubernetes devono essere in grado di montare i volumi di cui si è eseguito il provisioning per i pod. Se si utilizza `ontap-nas`, `ontap-nas-economy`, o. `ontap-nas-flexgroup` Driver per uno dei tuoi back-end, i nodi di lavoro hanno bisogno degli strumenti NFS. In caso contrario, richiedono gli strumenti iSCSI.

Le versioni recenti di RedHat CoreOS hanno sia NFS che iSCSI installati per impostazione predefinita.

> È necessario riavviare sempre i nodi di lavoro dopo l'installazione degli strumenti NFS o iSCSI, altrimenti il collegamento dei volumi ai container potrebbe non riuscire.

Volumi NFS

Protocollo	Sistema operativo	Comandi
NFS	RHEL/CentOS	`sudo yum install -y nfs-utils`
NFS	Ubuntu/Debian	`sudo apt-get install -y nfs-common`

> Assicurarsi che il servizio NFS venga avviato durante l'avvio.

Volumi iSCSI

Quando si utilizzano volumi iSCSI, considerare quanto segue:

- Ogni nodo del cluster Kubernetes deve avere un IQN univoco. **Questo è un prerequisito necessario**.
- Se si utilizza RHCOS versione 4.5 o successiva oppure RHEL o CentOS versione 8.2 o successiva con `solidfire-san` Verificare che l'algoritmo di autenticazione CHAP sia impostato su MD5 in `/etc/iscsi/iscsid.conf`.

```
sudo sed -i 's/^\(node.session.auth.chap_algs\).*/\1 = MD5/'
/etc/iscsi/iscsid.conf
```

- Quando si utilizzano nodi di lavoro che eseguono RHEL/RedHat CoreOS con iSCSI PVS, assicurarsi di specificare `discard` MountOption in StorageClass per eseguire la rigenerazione dello spazio inline. Vedere "La documentazione di RedHat".

Protocollo	Sistema operativo	Comandi

ISCSI	RHEL/CentOS	1. Installare i seguenti pacchetti di sistema:
		
		`sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-mapper-multipath`
		
		2. Verificare che la versione di iscsi-initiator-utils sia 6.2.0.874-2.el7 o successiva:
		
		`rpm -q iscsi-initiator-utils`
		
		3. Impostare la scansione su manuale:
		
		`sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/' /etc/iscsi/iscsid.conf`
		
		4. Abilitare il multipathing:
		
		`sudo mpathconf --enable --with_multipathd y --find_multipaths n`
		
		Assicurarsi `etc/multipath.conf` contiene `find_multipaths no` sotto `defaults`.
		
		5. Assicurarsi che `iscsid` e. `multipathd` sono in esecuzione:
		
		`sudo systemctl enable --now iscsid multipathd`
		
		6. Attivare e avviare `iscsi`:
		
		`sudo systemctl enable --now iscsi`

Protocollo	Sistema operativo	Comandi

ISCSI	Ubuntu/Debian	1. Installare i seguenti pacchetti di sistema:
		
		`sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools scsitools`
		
		2. Verificare che la versione Open-iscsi sia 2.0.874-5ubuntu2.10 o successiva (per il bionico) o 2.0.874-7.1ubuntu6.1 o successiva (per il focale):
		
		`dpkg -l open-iscsi`
		
		3. Impostare la scansione su manuale:
		
		`sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/' /etc/iscsi/iscsid.conf`
		
		4. Abilitare il multipathing:
		
		`sudo tee /etc/multipath.conf <←'EOF'`
		`defaults {`
		`user_friendly_names yes`
		`find_multipaths no`
		`}`
		`EOF`
		`sudo systemctl enable --now multipath-tools.service`
		`sudo service multipath-tools restart`
		
		Assicurarsi `etc/multipath.conf` contiene `find_multipaths no` sotto `defaults`.
		
		5. Assicurarsi che `open-iscsi` e. `multipath-tools` sono abilitati e in esecuzione:

		`sudo systemctl status multipath-tools`

Per Ubuntu 18.04, è necessario rilevare le porte di destinazione con `iscsiadm` prima di iniziare `open-iscsi` Per avviare il daemon iSCSI. In alternativa, è possibile modificare `iscsi` servizio da avviare `iscsid` automaticamente.

```
sudo systemctl enable
--now open-
iscsi.service
sudo systemctl status
open-iscsi
```

Per ulteriori informazioni sulla preparazione automatica del nodo di lavoro, che è una funzionalità beta, vedere "qui".

Preparazione automatica del nodo di lavoro

Astra Trident può installare automaticamente il necessario `NFS` e. `iSCSI` Strumenti sui nodi presenti nel cluster Kubernetes. Si tratta di una funzionalità * beta* e non è prevista per* cluster di produzione. Attualmente, la funzionalità è disponibile per i nodi che eseguono **CentOS, RHEL e Ubuntu**.

Per questa funzionalità, Astra Trident include un nuovo flag di installazione: `--enable-node-prep` per le installazioni implementate con `tridentctl`. Per le implementazioni con l'operatore Trident, utilizzare l'opzione booleano `enableNodePrep`.

Il `--enable-node-prep` L'opzione di installazione indica ad Astra Trident di installare e assicurarsi che i pacchetti e/o i servizi NFS e iSCSI siano in esecuzione quando un volume viene montato su un nodo di lavoro. Si tratta di una funzionalità * beta* destinata all'utilizzo in ambienti di sviluppo/test **non qualificati** per l'utilizzo in produzione.

Quando il `--enable-node-prep` Flag incluso per le installazioni Astra Trident implementate con `tridentctl`, ecco cosa succede:

1. Nell'ambito dell'installazione, Astra Trident registra i nodi su cui viene eseguito.
2. Quando viene effettuata una richiesta di rimborso del volume persistente (PVC), Astra Trident crea un PV da uno dei backend gestiti.
3. L'utilizzo del PVC in un pod richiede ad Astra Trident di montare il volume sul nodo su cui viene eseguito il pod. Astra Trident tenta di installare le utility client NFS/iSCSI necessarie e di garantire che i servizi richiesti siano attivi. Questa operazione viene eseguita prima del montaggio del volume.

La preparazione di un nodo di lavoro viene eseguita una sola volta come parte del primo tentativo di montaggio di un volume. Tutti i successivi montaggi di volume dovrebbero avere successo, purché non vengano apportate modifiche all'esterno di Astra Trident `NFS` e. `iSCSI` utilità.

In questo modo, Astra Trident può garantire che tutti i nodi di un cluster Kubernetes dispongano delle utility necessarie per montare e collegare i volumi. Per i volumi NFS, la policy di esportazione dovrebbe anche consentire il montaggio del volume. Trident può gestire automaticamente le policy di esportazione per backend; in alternativa, gli utenti possono gestire le policy di esportazione fuori banda.

Monitorare Astra Trident

Astra Trident fornisce un set di endpoint di metriche Prometheus che è possibile utilizzare per monitorare le performance di Astra Trident.

Le metriche fornite da Astra Trident ti consentono di:

- Tieni sotto controllo lo stato di salute e la configurazione di Astra Trident. È possibile esaminare il successo delle operazioni e se è in grado di comunicare con i back-end come previsto.

- Esaminare le informazioni sull'utilizzo del back-end e comprendere il numero di volumi sottoposti a provisioning su un back-end, la quantità di spazio consumato e così via.
- Mantenere una mappatura della quantità di volumi forniti sui backend disponibili.
- Tenere traccia delle performance. Puoi dare un'occhiata a quanto tempo ci vuole per Astra Trident per comunicare con i back-end ed eseguire le operazioni.

> Per impostazione predefinita, le metriche di Trident sono esposte sulla porta di destinazione `8001` su `/metrics` endpoint. Queste metriche sono **abilitate per impostazione predefinita** quando Trident è installato.

Di cosa hai bisogno

- Un cluster Kubernetes con Astra Trident installato.
- Un'istanza Prometheus. Questo può essere un "Implementazione di Prometheus in container" Oppure puoi scegliere di eseguire Prometheus come a. "applicazione nativa".

Fase 1: Definire un target Prometheus

Devi definire un target Prometheus per raccogliere le metriche e ottenere informazioni sui backend gestiti da Astra Trident, sui volumi creati e così via. Questo "blog" Spiega come utilizzare Prometheus e Grafana con Astra Trident per recuperare le metriche. Il blog spiega come eseguire Prometheus come operatore nel cluster Kubernetes e come creare un ServiceMonitor per ottenere le metriche di Astra Trident.

Fase 2: Creazione di un ServiceMonitor Prometheus

Per utilizzare le metriche Trident, è necessario creare un ServiceMonitor Prometheus che controlli `trident-csi` e ascolta su `metrics` porta. Un esempio di ServiceMonitor è simile al seguente:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
      release: prom-operator
  spec:
    jobLabel: trident
    selector:
      matchLabels:
        app: controller.csi.trident.netapp.io
    namespaceSelector:
      matchNames:
      - trident
    endpoints:
    - port: metrics
      interval: 15s
```

Questa definizione di ServiceMonitor recupera le metriche restituite da `trident-csi` e in particolare cerca di

`metrics` endpoint del servizio. Di conseguenza, Prometheus è ora configurato per comprendere le metriche di Astra Trident.

Oltre alle metriche disponibili direttamente da Astra Trident, Kubelet ne espone molte `kubelet_volume_*` metriche tramite il proprio endpoint di metriche. Kubelet può fornire informazioni sui volumi collegati, sui pod e sulle altre operazioni interne gestite. Vedere "qui".

Fase 3: Eseguire una query sulle metriche di Trident con PromQL

PromQL è utile per la creazione di espressioni che restituiscono dati di serie temporali o tabulari.

Di seguito sono riportate alcune query PromQL che è possibile utilizzare:

Ottieni informazioni sulla salute di Trident

- **Percentuale di risposte HTTP 2XX da Astra Trident**

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Percentuale di risposte REST da Astra Trident tramite codice di stato**

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Durata media in ms delle operazioni eseguite da Astra Trident**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Ottieni informazioni sull'utilizzo di Astra Trident

- **Dimensione media del volume**

```
trident_volume_allocated_bytes/trident_volume_count
```

- **Spazio totale del volume fornito da ciascun backend**

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

Ottieni l'utilizzo di singoli volumi

> Questa opzione è attivata solo se vengono raccolte anche le metriche del kubelet.

- **Percentuale di spazio utilizzato per ciascun volume**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Scopri di più sulla telemetria Astra Trident AutoSupport

Per impostazione predefinita, Astra Trident invia le metriche Prometheus e le informazioni di back-end di base a NetApp ogni giorno.

- Per impedire ad Astra Trident di inviare a NetApp le metriche Prometheus e le informazioni di back-end di base, passare il `--silence-autosupport` Segnalazione durante l'installazione di Astra Trident.
- Astra Trident può anche inviare i log dei container al NetApp Support on-demand tramite `tridentctl send autosupport`. Devi attivare Astra Trident per caricare i registri. Prima di inviare i log, è necessario accettare i file NetApphttps://www.netapp.com/company/legal/privacy-policy/["direttiva sulla privacy"^].
- Se non specificato, Astra Trident recupera i registri delle ultime 24 ore.
- È possibile specificare il periodo di conservazione dei log con `--since` allarme. Ad esempio: `tridentctl send autosupport --since=1h`. Queste informazioni vengono raccolte e inviate tramite un `trident-autosupport` Container installato insieme ad Astra Trident. È possibile ottenere l'immagine del contenitore in "Trident AutoSupport".
- Trident AutoSupport non raccoglie né trasmette dati personali o di identificazione personale (PII). Viene fornito con un "EULA" Non applicabile all'immagine del contenitore Trident. Scopri di più sull'impegno di NetApp per la sicurezza e la fiducia dei dati "qui".

Un payload di esempio inviato da Astra Trident è simile al seguente:

```
{
  "items": [
    {
      "backendUUID": "ff3852e1-18a5-4df4-b2d3-f59f829627ed",
      "protocol": "file",
      "config": {
        "version": 1,
        "storageDriverName": "ontap-nas",
        "debug": false,
        "debugTraceFlags": null,
        "disableDelete": false,
        "serialNumbers": [
          "nwkvzfanek_SN"
        ],
        "limitVolumeSize": ""
      },
      "state": "online",
      "online": true
    }
  ]
}
```


- I messaggi AutoSupport vengono inviati all'endpoint AutoSupport di NetApp. Se si utilizza un registro privato per memorizzare le immagini container, è possibile utilizzare `--image-registry` allarme.
- È inoltre possibile configurare gli URL proxy generando i file YAML di installazione. Per eseguire questa operazione, utilizzare `tridentctl install --generate-custom-yaml` Per creare i file YAML e aggiungere `--proxy-url` argomento per `trident-autosupport` container in `trident-deployment.yaml`.

Disattiva le metriche di Astra Trident

Per **disattivare** il report delle metriche, è necessario generare YAML personalizzati (utilizzando il `--generate-custom-yaml` e modificarli per rimuovere `--metrics` il contrassegno di non essere richiamato per `trident-main`container.

Astra Trident per Docker

Prerequisiti per l'implementazione

È necessario installare e configurare i prerequisiti del protocollo necessari sull'host prima di poter implementare Astra Trident.

- Verificare che l'implementazione soddisfi tutti i requisiti di "requisiti".
- Verificare che sia installata una versione supportata di Docker. Se la versione di Docker non è aggiornata, "installarlo o aggiornarlo".

```
docker --version
```

- Verificare che i prerequisiti del protocollo siano installati e configurati sull'host:

Protocollo	Sistema operativo	Comandi
NFS	RHEL/CentOS	`sudo yum install -y nfs-utils`
NFS	Ubuntu/Debian	`sudo apt-get install -y nfs-common`

Protocollo	Sistema operativo	Comandi

ISCSI	RHEL/CentOS 7	1. Installare i seguenti pacchetti di sistema:
		
		`sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-mapper-multipath`
		
		2. Verificare che la versione di iscsi-initiator-utils sia 6.2.0.874-2.el7 o successiva:
		
		`rpm -q iscsi-initiator-utils`
		
		3. Impostare la scansione su manuale:
		
		`sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/' /etc/iscsi/iscsid.conf`
		
		4. Abilitare il multipathing:
		
		`sudo mpathconf --enable --with_multipathd y --find_multipaths n`
		
		Assicurarsi `etc/multipath.conf` contiene `find_multipaths no` sotto `defaults`.
		
		5. Assicurarsi che `iscsid` e. `multipathd` sono in esecuzione:
		
		`sudo systemctl enable --now iscsid multipathd`
		
		6. Attivare e avviare `iscsi`:
		
		`sudo systemctl enable --now iscsi`

Protocollo	Sistema operativo	Comandi

ISCSI	Ubuntu	1. Installare i seguenti pacchetti di sistema:
		
		`sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools scsitools`
		
		2. Verificare che la versione Open-iscsi sia 2.0.874-5ubuntu2.10 o successiva (per il bionico) o 2.0.874-7.1ubuntu6.1 o successiva (per il focale):
		
		`dpkg -l open-iscsi`
		
		3. Impostare la scansione su manuale:
		
		`sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/' /etc/iscsi/iscsid.conf`
		
		4. Abilitare il multipathing:
		
		`sudo tee /etc/multipath.conf <←'EOF'`
		`defaults {`
		`user_friendly_names yes`
		`find_multipaths no`
		`}`
		`EOF`
		`sudo systemctl enable --now multipath-tools.service`
		`sudo service multipath-tools restart`
		
		Assicurarsi `etc/multipath.conf` contiene `find_multipaths no` sotto `defaults`.
		
		5. Assicurarsi che `open-iscsi` e. `multipath-tools` sono abilitati e in esecuzione:

		`sudo systemctl status multipath-tools`

Implementare Astra Trident

Astra Trident per Docker offre un'integrazione diretta con l'ecosistema Docker per le piattaforme storage NetApp. Supporta il provisioning e la gestione delle risorse di storage dalla piattaforma di storage agli host Docker, con un framework per aggiungere altre piattaforme in futuro.

Più istanze di Astra Trident possono essere eseguite contemporaneamente sullo stesso host. Ciò consente connessioni simultanee a più sistemi di storage e tipi di storage, con l'abilità di personalizzare lo storage utilizzato per i volumi Docker.

```
sudo systemctl enable
--now open-
iscsi.service
sudo systemctl status
open-iscsi
```

Di cosa hai bisogno

Vedere "prerequisiti per l'implementazione". Una volta soddisfatti i prerequisiti, è possibile implementare Astra Trident.

Metodo del plugin gestito da Docker (versione 1.13/17.03 e successive)

> **Prima di iniziare**
>
> Se hai utilizzato Astra Trident pre Docker 1.13/17.03 nel metodo daemon tradizionale, assicurati di arrestare il processo Astra Trident e riavviare il daemon Docker prima di utilizzare il metodo del plugin gestito.

1. Arrestare tutte le istanze in esecuzione:

```
pkill /usr/local/bin/netappdvp
pkill /usr/local/bin/trident
```

2. Riavviare Docker.

```
systemctl restart docker
```

3. Assicurarsi di avere installato Docker Engine 17.03 (nuovo 1.13) o versione successiva.

```
docker --version
```

   Se la versione non è aggiornata, "installare o aggiornare l'installazione".

Fasi

1. Creare un file di configurazione e specificare le opzioni come segue:
   - `config`: Il nome file predefinito è `config.json`, tuttavia, è possibile utilizzare qualsiasi nome scegliendo specificando il `config` con il nome del file. Il file di configurazione deve trovarsi in `/etc/netappdvp` directory sul sistema host.
   - `log-level`: Specificare il livello di registrazione (`debug`, `info`, `warn`, `error`, `fatal`). L'impostazione predefinita è `info`.
   - `debug`: Specificare se la registrazione di debug è attivata. Il valore predefinito è false. Sovrascrive log-level se true.

i. Creare un percorso per il file di configurazione:

```
sudo mkdir -p /etc/netappdvp
```

ii. Creare il file di configurazione:

```
cat << EOF > /etc/netappdvp/config.json
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",
    "aggregate": "aggr1"
}
EOF
```

2. Avviare Astra Trident utilizzando il sistema di plugin gestito.

```
docker plugin install --grant-all-permissions --alias netapp
netapp/trident-plugin:21.07 config=myConfigFile.json
```

3. Iniziare a utilizzare Astra Trident per consumare lo storage dal sistema configurato.
   a. Creare un volume denominato "firstVolume":

```
docker volume create -d netapp --name firstVolume
```

   b. Creare un volume predefinito all'avvio del container:

```
docker run --rm -it --volume-driver netapp --volume
secondVolume:/my_vol alpine ash
```

   c. Rimuovere il volume "firstVolume":

```
docker volume rm firstVolume
```

Metodo tradizionale (versione 1.12 o precedente)

Prima di iniziare

1. Assicurarsi di disporre di Docker versione 1.10 o successiva.

   ```
   docker --version
   ```

   Se la versione non è aggiornata, aggiornare l'installazione.

   ```
   curl -fsSL https://get.docker.com/ | sh
   ```

   Oppure "seguire le istruzioni per la distribuzione".

2. Assicurarsi che NFS e/o iSCSI siano configurati per il sistema.

Fasi

1. Installare e configurare il plug-in NetApp Docker Volume:
   a. Scaricare e disimballare l'applicazione:

      ```
      wget
      https://github.com/NetApp/trident/releases/download/v21.04.0/trident-
      installer-21.07.0.tar.gz
      tar zxf trident-installer-21.07.0.tar.gz
      ```

   b. Spostarsi in una posizione nel percorso del vassoio:

      ```
      sudo mv trident-installer/extras/bin/trident /usr/local/bin/
      sudo chown root:root /usr/local/bin/trident
      sudo chmod 755 /usr/local/bin/trident
      ```

   c. Creare un percorso per il file di configurazione:

      ```
      sudo mkdir -p /etc/netappdvp
      ```

   d. Creare il file di configurazione:

```
cat << EOF > /etc/netappdvp/ontap-nas.json
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",
    "aggregate": "aggr1"
}
EOF
```

2. Dopo aver posizionato il file binario e aver creato i file di configurazione, avviare il daemon Trident utilizzando il file di configurazione desiderato.

```
sudo trident --config=/etc/netappdvp/ontap-nas.json
```

> Se non specificato, il nome predefinito per il driver del volume è "netapp".

Una volta avviato il daemon, è possibile creare e gestire i volumi utilizzando l'interfaccia CLI di Docker

3. Creare un volume:

```
docker volume create -d netapp --name trident_1
```

4. Provisioning di un volume Docker all'avvio di un container:

```
docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol
alpine ash
```

5. Rimuovere un volume Docker:

```
docker volume rm trident_1
docker volume rm trident_2
```

Avviare Astra Trident all'avvio del sistema

Un file di unità di esempio per i sistemi basati su sistema è disponibile all'indirizzo `contrib/trident.service.example` Nel Git repo. Per utilizzare il file con CentOS/RHEL, procedere come segue:

1. Copiare il file nella posizione corretta.

   Se sono in esecuzione più istanze, utilizzare nomi univoci per i file di unità.

   ```
   cp contrib/trident.service.example
   /usr/lib/systemd/system/trident.service
   ```

2. Modificare il file, modificare la descrizione (riga 2) in modo che corrisponda al nome del driver e al percorso del file di configurazione (riga 9) in base all'ambiente in uso.
3. Ricaricare il sistema per l'IT per acquisire le modifiche:

   ```
   systemctl daemon-reload
   ```

4. Attivare il servizio.

   Questo nome varia in base al nome del file in `/usr/lib/systemd/system` directory.

   ```
   systemctl enable trident
   ```

5. Avviare il servizio.

   ```
   systemctl start trident
   ```

6. Visualizzare lo stato.

   ```
   systemctl status trident
   ```

> Ogni volta che si modifica il file di unità, eseguire `systemctl daemon-reload` per essere consapevole delle modifiche.

Aggiornare o disinstallare Astra Trident

Puoi aggiornare Astra Trident per Docker senza alcun impatto sui volumi in uso. Durante il processo di aggiornamento, ci sarà un breve periodo in cui `docker volume` i comandi diretti al plugin non avranno successo e le applicazioni non potranno montare i volumi fino a quando il plugin non verrà eseguito di nuovo. Nella maggior parte dei casi, si tratta di pochi secondi.

Eseguire l'upgrade

Per aggiornare Astra Trident per Docker, attenersi alla procedura riportata di seguito.

Fasi

1. Elencare i volumi esistenti:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

2. Disattivare il plug-in:

```
docker plugin disable -f netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Aggiornare il plug-in:

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```

> La release 18.01 di Astra Trident sostituisce l'nDVP. È necessario eseguire l'aggiornamento direttamente da `netapp/ndvp-plugin` al `netapp/trident-plugin` immagine.

4. Attivare il plug-in:

```
docker plugin enable netapp:latest
```

5. Verificare che il plug-in sia attivato:

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       Trident - NetApp Docker Volume
Plugin   true
```

6. Verificare che i volumi siano visibili:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

> Se si esegue l'aggiornamento da una versione precedente di Astra Trident (precedente alla 20.10) ad Astra Trident 20.10 o successiva, potrebbe verificarsi un errore. Per ulteriori informazioni, vedere "Problemi noti". Se si verifica l'errore, disattivare il plug-in, quindi rimuovere il plug-in e installare la versione richiesta di Astra Trident passando un parametro di configurazione aggiuntivo: `docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all-permissions config=config.json`

Disinstallare

Per disinstallare Astra Trident per Docker, procedere come segue.

Fasi

1. Rimuovere tutti i volumi creati dal plug-in.
2. Disattivare il plug-in:

```
docker plugin disable netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Rimuovere il plug-in:

```
docker plugin rm netapp:latest
```

Lavorare con i volumi

È possibile creare, clonare e rimuovere facilmente i volumi utilizzando lo standard `docker volume` Comandi con il nome del driver Astra Trident specificato quando necessario.

Creare un volume

- Creare un volume con un driver utilizzando il nome predefinito:

```
docker volume create -d netapp --name firstVolume
```

- Creare un volume con un'istanza specifica di Astra Trident:

```
docker volume create -d ntap_bronze --name bronzeVolume
```

Se non si specifica alcuna "opzioni", vengono utilizzate le impostazioni predefinite del driver.

- Eseguire l'override delle dimensioni predefinite del volume. Per creare un volume 20GiB con un driver, vedere l'esempio seguente:

```
docker volume create -d netapp --name my_vol --opt size=20G
```

Le dimensioni dei volumi sono espresse come stringhe contenenti un valore intero con unità opzionali (ad esempio 10G, 20GB, 3TiB). Se non viene specificata alcuna unità, l'impostazione predefinita è G. Le unità di misura possono essere espresse come potenze di 2 (B, KiB, MiB, GiB, TIB) o potenze di 10 (B, KB, MB, GB, TB). Le unità shortand utilizzano potenze di 2 (G = GiB, T = TIB, …).

Rimuovere un volume

- Rimuovere il volume come qualsiasi altro volume Docker:

```
docker volume rm firstVolume
```

Quando si utilizza `solidfire-san` driver, l'esempio precedente elimina e cancella il volume.

Per aggiornare Astra Trident per Docker, attenersi alla procedura riportata di seguito.

Clonare un volume

Quando si utilizza `ontap-nas`, `ontap-san`, `solidfire-san`, e. `gcp-cvs storage drivers`, Astra Trident può clonare i volumi. Quando si utilizza `ontap-nas-flexgroup` oppure `ontap-nas-economy` driver, la clonazione non è supportata. La creazione di un nuovo volume da un volume esistente determinerà la creazione di un nuovo snapshot.

- Esaminare il volume per enumerare gli snapshot:

```
docker volume inspect <volume_name>
```

- Creare un nuovo volume da un volume esistente. In questo modo verrà creata una nuova istantanea:

```
docker volume create -d <driver_name> --name <new_name> -o
from=<source_docker_volume>
```

- Creare un nuovo volume da uno snapshot esistente su un volume. In questo modo non viene creata una nuova istantanea:

```
docker volume create -d <driver_name> --name <new_name> -o
from=<source_docker_volume> -o fromSnapshot=<source_snap_name>
```

Esempio

```
[me@host ~]$ docker volume inspect firstVolume

[
    {
        "Driver": "ontap-nas",
        "Labels": null,
        "Mountpoint": "/var/lib/docker-volumes/ontap-
nas/netappdvp_firstVolume",
        "Name": "firstVolume",
        "Options": {},
        "Scope": "global",
        "Status": {
            "Snapshots": [
                {
                    "Created": "2017-02-10T19:05:00Z",
                    "Name": "hourly.2017-02-10_1505"
                }
            ]
        }
    }
]

[me@host ~]$ docker volume create -d ontap-nas --name clonedVolume -o
from=firstVolume
clonedVolume

[me@host ~]$ docker volume rm clonedVolume
[me@host ~]$ docker volume create -d ontap-nas --name volFromSnap -o
from=firstVolume -o fromSnapshot=hourly.2017-02-10_1505
volFromSnap

[me@host ~]$ docker volume rm volFromSnap
```

Accesso ai volumi creati esternamente

È possibile accedere ai dispositivi a blocchi creati esternamente (o ai loro cloni) dai container usando Trident **solo** se non hanno partizioni e se il loro filesystem è supportato da Astra Trident (ad esempio: An `ext4`formattato `/dev/sdc1` Non sarà accessibile tramite Astra Trident).

Opzioni di volume specifiche del driver

Ciascun driver di storage dispone di un set di opzioni diverso, che è possibile specificare al momento della creazione del volume per personalizzare il risultato. Di seguito sono riportate le opzioni applicabili al sistema di storage configurato.

L'utilizzo di queste opzioni durante l'operazione di creazione del volume è semplice. Fornire l'opzione e il valore utilizzando `-o` Durante l'operazione CLI. Questi valori sovrascrivono qualsiasi valore equivalente dal file di configurazione JSON.

Opzioni del volume ONTAP

Le opzioni di creazione dei volumi per NFS e iSCSI includono quanto segue:

Opzione	Descrizione
`size`	La dimensione predefinita del volume è 1 GiB.
`spaceReserve`	Thin provisioning o thick provisioning del volume, per impostazione predefinita thin. I valori validi sono `none` (con thin provisioning) e. `volume` (thick provisioning).
`snapshotPolicy`	In questo modo, il criterio di snapshot viene impostato sul valore desiderato. L'impostazione predefinita è `none`, ovvero non verranno create automaticamente snapshot per il volume. A meno che non venga modificato dall'amministratore dello storage, su tutti i sistemi ONTAP esiste una policy denominata "default" che crea e conserva sei snapshot ogni ora, due giornalieri e due settimanali. I dati conservati in uno snapshot possono essere ripristinati esplorando il `.snapshot` directory in qualsiasi directory del volume.
`snapshotReserve`	In questo modo si imposta la riserva di snapshot sulla percentuale desiderata. Il valore predefinito è NO, ovvero ONTAP selezionerà snapshotReserve (di solito 5%) se è stata selezionata una snapshotPolicy, o 0% se la snapshotPolicy non è nessuna. È possibile impostare il valore predefinito snapshotReserve nel file di configurazione per tutti i backend ONTAP e utilizzarlo come opzione di creazione di volumi per tutti i backend ONTAP ad eccezione di ontap-nas-Economy.

Opzione	Descrizione
`splitOnClone`	Durante il cloning di un volume, ONTAP suddividerà immediatamente il clone dal suo padre. L'impostazione predefinita è `false`. Alcuni casi di utilizzo per il cloning dei volumi sono meglio serviti dalla suddivisione del clone dal suo padre immediatamente dopo la creazione, perché è improbabile che vi siano opportunità di efficienza dello storage. Ad esempio, il cloning di un database vuoto può offrire notevoli risparmi di tempo, ma anche uno storage ridotto, pertanto è meglio suddividere immediatamente il clone.
`encryption`	In questo modo, NetApp Volume Encryption (NVE) verrà attivato sul nuovo volume, per impostazione predefinita a. `false`. NVE deve essere concesso in licenza e abilitato sul cluster per utilizzare questa opzione. NetApp aggregate Encryption (NAE) non è attualmente supportato in Trident.
`tieringPolicy`	Imposta il criterio di tiering da utilizzare per il volume. In questo modo si decide se i dati vengono spostati nel livello cloud quando diventano inattivi (freddo).

Le seguenti opzioni aggiuntive sono per NFS **only**:

Opzione	Descrizione
`unixPermissions`	In questo modo viene controllato il set di autorizzazioni per il volume stesso. Per impostazione predefinita, le autorizzazioni vengono impostate su `` `---rwxr-xr-x ``, o nella notazione numerica 0755, e. `root` sarà il proprietario. Il formato di testo o numerico funziona.
`snapshotDir`	Impostare questa opzione su `true` farà il `.snapshot` directory visibile ai client che accedono al volume. Il valore predefinito è `false`, il che significa che la visibilità di `.snapshot` la directory è disattivata per impostazione predefinita. Alcune immagini, ad esempio l'immagine ufficiale di MySQL, non funzionano come previsto quando `.snapshot` la directory è visibile.
`exportPolicy`	Imposta il criterio di esportazione da utilizzare per il volume. L'impostazione predefinita è `default`.

Opzione	Descrizione
`securityStyle`	Imposta lo stile di sicurezza da utilizzare per l'accesso al volume. L'impostazione predefinita è `unix`. I valori validi sono `unix` e. `mixed`.

Le seguenti opzioni aggiuntive sono disponibili solo per iSCSI*:

Opzione	Descrizione
`fileSystemType`	Imposta il file system utilizzato per formattare i volumi iSCSI. L'impostazione predefinita è `ext4`. I valori validi sono `ext3`, `ext4`, e. `xfs`.
`spaceAllocation`	Impostare questa opzione su `false` Disattiva la funzione di allocazione dello spazio del LUN. Il valore predefinito è `true`, Ovvero ONTAP notifica all'host quando il volume ha esaurito lo spazio e il LUN nel volume non può accettare le scritture. Questa opzione consente inoltre a ONTAP di recuperare automaticamente lo spazio quando l'host elimina i dati.

Esempi

Vedere gli esempi riportati di seguito:

- Creazione di un volume da 10 GiB:

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- Creazione di un volume 100GiB con snapshot:

```
docker volume create -d netapp --name demo -o size=100G -o
snapshotPolicy=default -o snapshotReserve=10
```

- Creare un volume con il bit setuid attivato:

```
docker volume create -d netapp --name demo -o unixPermissions=4755
```

Le dimensioni minime del volume sono 20 MiB.

Se la riserva di snapshot non viene specificata e la policy di snapshot è `none`, Trident utilizzerà una riserva di snapshot dello 0%.

- Creare un volume senza policy di snapshot e senza riserva di snapshot:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- Creare un volume senza policy di snapshot e una riserva di snapshot personalizzata del 10%:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- Creare un volume con una policy di snapshot e una riserva di snapshot personalizzata del 10%:

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy --opt snapshotReserve=10
```

- Creare un volume con una policy di snapshot e accettare la riserva di snapshot predefinita di ONTAP (di solito il 5%):

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy
```

Opzioni volume software Element

Le opzioni del software Element espongono le dimensioni e i criteri di qualità del servizio (QoS) associati al volume. Una volta creato il volume, il criterio QoS associato viene specificato utilizzando `-o type=service_level` nomenclatura.

Il primo passo per definire un livello di servizio QoS con il driver Element consiste nel creare almeno un tipo e specificare gli IOPS minimi, massimi e burst associati a un nome nel file di configurazione.

Le altre opzioni di creazione dei volumi software Element includono:

Opzione	Descrizione
`size`	La dimensione del volume, per impostazione predefinita è 1GiB o voce di configurazione … "Default": {"size": "5G"}.
`blocksize`	Utilizzare 512 o 4096, il valore predefinito è 512 o la voce di configurazione DefaultBlockSize.

Esempio

Vedere il seguente file di configurazione di esempio con le definizioni di QoS:

```
{
    "...": "..."
    "Types": [
        {
            "Type": "Bronze",
            "Qos": {
                "minIOPS": 1000,
                "maxIOPS": 2000,
                "burstIOPS": 4000
            }
        },
        {
            "Type": "Silver",
            "Qos": {
                "minIOPS": 4000,
                "maxIOPS": 6000,
                "burstIOPS": 8000
            }
        },
        {
            "Type": "Gold",
            "Qos": {
                "minIOPS": 6000,
                "maxIOPS": 8000,
                "burstIOPS": 10000
            }
        }
    ]
}
```

Nella configurazione precedente, sono disponibili tre definizioni di policy: Bronze, Silver e Gold. Questi nomi sono arbitrari.

- Crea un volume Gold da 10 GiB:

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- Crea un volume Bronze da 100 GiB:

```
docker volume create -d solidfire --name sfBronze -o type=Bronze -o
size=100G
```

CVS sulle opzioni del volume GCP

Le opzioni di creazione dei volumi per il driver CVS su GCP includono:

Opzione	Descrizione
`size`	Le dimensioni del volume, per impostazione predefinita, sono 100 GiB per i volumi CVS-Performance o 300 GiB per i volumi CVS.
`serviceLevel`	Il livello di servizio CVS del volume, per impostazione predefinita, è standard. I valori validi sono standard, premium ed estremi.
`snapshotReserve`	In questo modo si imposta la riserva di snapshot sulla percentuale desiderata. Il valore predefinito è NO, ovvero CVS selezionerà la riserva di snapshot (di solito 0%).

Esempi

- Creare un volume 2TiB:

```
docker volume create -d netapp --name demo -o size=2T
```

- Crea un volume premium 5TiB:

```
docker volume create -d netapp --name demo -o size=5T -o
serviceLevel=premium
```

La dimensione minima del volume è 100 GiB per i volumi CVS-Performance o 300 GiB per i volumi CVS.

Opzioni del volume Azure NetApp Files

Le opzioni di creazione dei volumi per il driver Azure NetApp Files includono:

Opzione	Descrizione
`size`	La dimensione predefinita del volume è 100 GB.

Esempi

- Creare un volume 200GiB:

```
docker volume create -d netapp --name demo -o size=200G
```

Le dimensioni minime del volume sono di 100 GB.

Raccogliere i log

È possibile raccogliere i registri per ottenere assistenza nella risoluzione dei problemi. Il metodo utilizzato per raccogliere i log varia in base alla modalità di esecuzione del plug-in Docker.

Fasi

1. Se si esegue Astra Trident utilizzando il metodo di plugin gestito consigliato (ad esempio, utilizzando `docker plugin` e visualizzarli come segue:

   ```
   # docker plugin ls
   ID                  NAME                DESCRIPTION
   ENABLED
   4fb97d2b956b        netapp:latest       nDVP - NetApp Docker Volume
   Plugin   false
   # journalctl -u docker | grep 4fb97d2b956b
   ```

   Il livello di registrazione standard dovrebbe consentire di diagnosticare la maggior parte dei problemi. Se non è sufficiente, è possibile attivare la registrazione del debug.

2. Per abilitare la registrazione del debug, installare il plug-in con la registrazione del debug attivata:

   ```
   docker plugin install netapp/trident-plugin:<version> --alias <alias>
   debug=true
   ```

   In alternativa, attivare la registrazione del debug quando il plug-in è già installato:

   ```
   docker plugin disable <plugin>
   docker plugin set <plugin> debug=true
   docker plugin enable <plugin>
   ```

3. Se si esegue il file binario sull'host, i registri sono disponibili in quello dell'host `/var/log/netappdvp` directory. Per attivare la registrazione di debug, specificare `-debug` quando si esegue il plug-in.

Suggerimenti generali per la risoluzione dei problemi

- Il problema più comune in cui i nuovi utenti eseguono è una configurazione errata che impedisce l'inizializzazione del plug-in. In questo caso, quando si tenta di installare o abilitare il plug-in, viene visualizzato un messaggio simile al seguente:

  ```
  Error response from daemon: dial unix /run/docker/plugins/<id>/netapp.sock:
  connect: no such file or directory
  ```

  Ciò significa che il plug-in non è stato avviato. Fortunatamente, il plug-in è stato creato con una funzionalità di registrazione completa che dovrebbe aiutarti a diagnosticare la maggior parte dei problemi che probabilmente si verificano.

- In caso di problemi con il montaggio di un PV su un container, assicurarsi che `rpcbind` è installato e in esecuzione. Utilizzare il gestore dei pacchetti richiesto per il sistema operativo host e verificare se `rpcbind` è in esecuzione. È possibile controllare lo stato del servizio rpcbind eseguendo un `systemctl status rpcbind` o equivalente.

Gestire più istanze di Astra Trident

Sono necessarie più istanze di Trident quando si desidera avere più configurazioni di storage disponibili contemporaneamente. La chiave per più istanze è assegnare loro nomi diversi utilizzando `--alias` con il plug-in containerizzato, o. `--volume-driver` Opzione durante l'istanza di Trident sull'host.

Procedura per il plug-in gestito da Docker (versione 1.13/17.03 o successiva)

1. Avviare la prima istanza specificando un alias e un file di configurazione.

```
docker plugin install --grant-all-permissions --alias silver
netapp/trident-plugin:21.07 config=silver.json
```

2. Avviare la seconda istanza, specificando un alias e un file di configurazione diversi.

```
docker plugin install --grant-all-permissions --alias gold
netapp/trident-plugin:21.07 config=gold.json
```

3. Creare volumi specificando l'alias come nome del driver.

   Ad esempio, per il volume gold:

```
docker volume create -d gold --name ntapGold
```

   Ad esempio, per il volume Silver:

```
docker volume create -d silver --name ntapSilver
```

Procedura per la versione tradizionale (1.12 o precedente)

1. Avviare il plug-in con una configurazione NFS utilizzando un ID driver personalizzato:

```
sudo trident --volume-driver=netapp-nas --config=/path/to/config
-nfs.json
```

2. Avviare il plug-in con una configurazione iSCSI utilizzando un ID driver personalizzato:

```
sudo trident --volume-driver=netapp-san --config=/path/to/config
-iscsi.json
```

3. Provisioning dei volumi Docker per ogni istanza del driver:

   Ad esempio, per NFS:

```
docker volume create -d netapp-nas --name my_nfs_vol
```

   Ad esempio, per iSCSI:

```
docker volume create -d netapp-san --name my_iscsi_vol
```

Opzioni di configurazione dello storage

Consulta le opzioni di configurazione disponibili per le configurazioni di Astra Trident.

Opzioni di configurazione globale

Queste opzioni di configurazione si applicano a tutte le configurazioni Astra Trident, indipendentemente dalla piattaforma di storage utilizzata.

Opzione	Descrizione	Esempio
`version`	Numero di versione del file di configurazione	1
`storageDriverName`	Nome del driver di storage	`ontap-nas`, `ontap-san`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `solidfire-san`, `azure-netapp-files`, o. `gcp-cvs`
`storagePrefix`	Prefisso opzionale per i nomi dei volumi. Impostazione predefinita: "Netappdvp_".	staging_
`limitVolumeSize`	Restrizione opzionale sulle dimensioni dei volumi. Predefinito: "" (non applicato)	10 g.

Non utilizzare `storagePrefix` (Incluso il valore predefinito) per i backend degli elementi. Per impostazione predefinita, il `solidfire-san` il driver ignora questa impostazione e non utilizza un prefisso. Si consiglia di utilizzare un tenantId specifico per la mappatura dei volumi Docker o i dati degli attributi che vengono popolati con la versione Docker, le informazioni sul driver e il nome raw di Docker nei casi in cui sia stato utilizzato il comando dei nomi.

Sono disponibili opzioni predefinite per evitare di doverle specificare su ogni volume creato. Il `size` l'opzione è disponibile per tutti i tipi di controller. Consultare la sezione relativa alla configurazione di ONTAP per un esempio su come impostare le dimensioni predefinite del volume.

Opzione	Descrizione	Esempio
`size`	Dimensione predefinita opzionale per i nuovi volumi. Predefinito: "1G"	10 G.

Configurazione di ONTAP

Oltre ai valori di configurazione globali sopra indicati, quando si utilizza ONTAP, sono disponibili le seguenti opzioni di primo livello.

Opzione	Descrizione	Esempio
`managementLIF`	Indirizzo IP della LIF di gestione ONTAP. È possibile specificare un nome di dominio completo (FQDN).	10.0.0.1
`dataLIF`	L'indirizzo IP del protocollo LIF; verrà derivato se non specificato. Per `ontap-nas` Driver **only**, è possibile specificare un FQDN, nel qual caso il FQDN verrà utilizzato per le operazioni di montaggio NFS. Per `ontap-san` Driver, l'impostazione predefinita prevede l'utilizzo di tutti gli IP LIF dei dati dalla SVM e l'utilizzo di multipath iSCSI. Specifica di un indirizzo IP per `dataLIF` per `ontap-san` i driver forzano il driver a disabilitare multipath e a utilizzare solo l'indirizzo specificato.	10.0.0.2
`svm`	Macchina virtuale per lo storage da utilizzare (obbligatorio, se la LIF di gestione è una LIF del cluster)	svm_nfs
`username`	Nome utente per la connessione al dispositivo di storage	vsadmin

Opzione	Descrizione	Esempio
`password`	Password per la connessione al dispositivo di storage	segreto
`aggregate`	Aggregato per il provisioning (facoltativo; se impostato, deve essere assegnato alla SVM). Per `ontap-nas-flexgroup` driver, questa opzione viene ignorata. Tutti gli aggregati assegnati alla SVM vengono utilizzati per il provisioning di un volume FlexGroup.	aggr1
`limitAggregateUsage`	Facoltativo, non eseguire il provisioning se l'utilizzo è superiore a questa percentuale	75%
`nfsMountOptions`	Controllo dettagliato delle opzioni di montaggio NFS; il valore predefinito è "-o nfsvers=3". **Disponibile solo per `ontap-nas` e. `ontap-nas-economy` driver**. "Fare clic qui per informazioni sulla configurazione degli host NFS".	-o nfsvers=4
`igroupName`	L'igroup utilizzato dal plugin; il default è "netappdvp". **Disponibile solo per il fiume `ontap-san`d**.	miigroup
`limitVolumeSize`	Dimensione massima del volume richiedente e dimensione massima del volume padre qtree. **Per `ontap-nas-economy` Driver, questa opzione limita inoltre le dimensioni dei FlexVol creati**.	300 g.
`qtreesPerFlexvol`	Il numero massimo di qtree per FlexVol deve essere compreso nell'intervallo [50, 300], il valore predefinito è 200. **Per `ontap-nas-economy` Driver, questa opzione consente di personalizzare il numero massimo di qtree per FlexVol**.	300

Sono disponibili opzioni predefinite per evitare di doverle specificare su ogni volume creato:

Opzione	Descrizione	Esempio
`spaceReserve`	Modalità di riserva dello spazio; "nessuno" (thin provisioning) o "volume" (thick)	nessuno
`snapshotPolicy`	Policy di Snapshot da utilizzare, il valore predefinito è "nessuno"	nessuno
`snapshotReserve`	Snapshot Reserve percent (percentuale riserva snapshot), il valore predefinito è "" per accettare il valore predefinito di ONTAP	10
`splitOnClone`	Dividere un clone dal suo padre al momento della creazione, per impostazione predefinita su "false"	falso
`encryption`	"Enable NetApp Volume Encryption (attiva crittografia volumi NetApp), il valore predefinito è "false"	vero
`unixPermissions`	Opzione NAS per volumi NFS con provisioning, valore predefinito "777"	777
`snapshotDir`	Opzione NAS per l'accesso a `.snapshot` directory, per impostazione predefinita impostata su "false"	vero
`exportPolicy`	Opzione NAS per l'utilizzo della policy di esportazione NFS, per impostazione predefinita "default"	predefinito
`securityStyle`	Opzione NAS per l'accesso al volume NFS fornito, per impostazione predefinita "unix"	misto
`fileSystemType`	OPZIONE SAN per selezionare il tipo di file system, il valore predefinito è "ext4"	xfs
`tieringPolicy`	Policy di tiering da utilizzare, il valore predefinito è "nessuno"; "solo snapshot" per la configurazione SVM-DR precedente a ONTAP 9.5	nessuno

Opzioni di scalabilità

Il `ontap-nas` e. `ontap-san` I driver creano un ONTAP FlexVol per ogni volume Docker. ONTAP supporta fino a 1000 FlexVol per nodo cluster con un massimo di 12,000 FlexVol. Se i requisiti del volume Docker rientrano in tale limite, il `ontap-nas` Il driver è la soluzione NAS preferita a causa delle funzionalità aggiuntive offerte da FlexVol, come le snapshot Docker-volume-granulare e la clonazione.

Se hai bisogno di più volumi Docker di quelli che possono essere contenuti nei limiti FlexVol, scegli `ontap-nas-economy` o il `ontap-san-economy` driver.

Il `ontap-nas-economy` Driver crea volumi Docker come Qtree ONTAP all'interno di un pool di FlexVol gestiti automaticamente. I qtree offrono una scalabilità di gran lunga superiore, fino a 100,000 per nodo cluster e 2,400,000 per cluster, a scapito di alcune funzionalità. Il `ontap-nas-economy` Il driver non supporta snapshot o cloning granulari dei volumi Docker.

> Il `ontap-nas-economy` Il driver non è attualmente supportato in Docker Swame, perché Swarm non orchestrava la creazione di volumi su più nodi.

Il `ontap-san-economy` Driver crea volumi Docker come LUN ONTAP all'interno di un pool condiviso di FlexVol gestiti automaticamente. In questo modo, ogni FlexVol non è limitato a un solo LUN e offre una migliore scalabilità per i carichi di lavoro SAN. A seconda dello storage array, ONTAP supporta fino a 16384 LUN per cluster. Poiché i volumi sono LUN sottostanti, questo driver supporta snapshot e cloning Docker-volume-granulare.

Scegliere `ontap-nas-flexgroup` il driver per aumentare il parallelismo a un singolo volume che può crescere nell'intervallo dei petabyte con miliardi di file. Alcuni casi di utilizzo ideali per FlexGroups includono ai/ML/DL, big data e analytics, build software, streaming, repository di file e così via. Trident utilizza tutti gli aggregati assegnati a una SVM durante il provisioning di un volume FlexGroup. Il supporto di FlexGroup in Trident ha anche le seguenti considerazioni:

- Richiede ONTAP versione 9.2 o successiva.
- Al momento della stesura del presente documento, FlexGroups supporta solo NFS v3.
- Si consiglia di attivare gli identificatori NFSv3 a 64 bit per SVM.
- La dimensione minima consigliata per il FlexGroup è di 100 GB.
- La clonazione non è supportata per i volumi FlexGroup.

Per informazioni su FlexGroups e workload appropriati per FlexGroups, vedere "Guida all'implementazione e alle Best practice per i volumi NetApp FlexGroup".

Per ottenere funzionalità avanzate e scalabilità enorme nello stesso ambiente, è possibile eseguire più istanze del Docker Volume Plugin, con una sola applicazione `ontap-nas` e un altro utilizzo `ontap-nas-economy`.

File di configurazione ONTAP di esempio

Esempio NFS per `ontap-nas` driver

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",
    "aggregate": "aggr1",
    "defaults": {
      "size": "10G",
      "spaceReserve": "none",
      "exportPolicy": "default"
    }
}
```

Esempio NFS per `ontap-nas-flexgroup` driver

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",
    "defaults": {
      "size": "100G",
      "spaceReserve": "none",
      "exportPolicy": "default"
    }
}
```

Esempio NFS per `ontap-nas-economy` driver

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",
    "aggregate": "aggr1"
}
```

Esempio iSCSI per `ontap-san` driver

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "username": "vsadmin",
    "password": "secret",
    "aggregate": "aggr1",
    "igroupName": "myigroup"
}
```

Esempio NFS per `ontap-san-economy` driver

```
{
    "version": 1,
    "storageDriverName": "ontap-san-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi_eco",
    "username": "vsadmin",
    "password": "secret",
    "aggregate": "aggr1",
    "igroupName": "myigroup"
}
```

Configurazione del software Element

Oltre ai valori di configurazione globali, quando si utilizza il software Element (NetApp HCI/SolidFire), queste opzioni sono disponibili.

Opzione	Descrizione	Esempio
`Endpoint`	https://<login>:<password>@<mvip>/json-rpc/<element-version>	https://admin:admin@192.168.160.3/json-rpc/8.0
`SVIP`	Porta e indirizzo IP iSCSI	10.0.0.7:3260
`TenantName`	Tenant SolidFireF da utilizzare (creato se non trovato)	"docker"
`InitiatorIFace`	Specificare l'interfaccia quando si limita il traffico iSCSI all'interfaccia non predefinita	"predefinito"
`Types`	Specifiche QoS	Vedere l'esempio riportato di seguito
`LegacyNamePrefix`	Prefisso per installazioni Trident aggiornate. Se è stata utilizzata una versione di Trident precedente alla 1.3.2 ed è stato eseguito un aggiornamento con i volumi esistenti, è necessario impostare questo valore per accedere ai volumi precedenti che sono stati mappati tramite il metodo del nome del volume.	"netappdvp-"

Il `solidfire-san` Il driver non supporta Docker Swarm.

Esempio di file di configurazione del software Element

```
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://admin:admin@192.168.160.3/json-rpc/8.0",
    "SVIP": "10.0.0.7:3260",
    "TenantName": "docker",
    "InitiatorIFace": "default",
    "Types": [
        {
            "Type": "Bronze",
            "Qos": {
                "minIOPS": 1000,
                "maxIOPS": 2000,
                "burstIOPS": 4000
            }
        },
        {
            "Type": "Silver",
            "Qos": {
                "minIOPS": 4000,
                "maxIOPS": 6000,
                "burstIOPS": 8000
            }
        },
        {
            "Type": "Gold",
            "Qos": {
                "minIOPS": 6000,
                "maxIOPS": 8000,
                "burstIOPS": 10000
            }
        }
    ]
}
```

Cloud Volumes Service (CVS) su configurazione GCP

Trident ora include il supporto per volumi più piccoli con il tipo di servizio CVS predefinito attivato "GCP". Per i backend creati con `storageClass=software`, I volumi avranno ora una dimensione di provisioning minima di 300 GiB. **NetApp consiglia ai clienti di consumare volumi inferiori a 1 TiB per carichi di lavoro non in produzione**. CVS attualmente fornisce questa funzione in disponibilità controllata e non fornisce supporto tecnico.

> Iscriviti per accedere a volumi inferiori a 1 TiB "qui".

> Quando si implementano backend utilizzando il tipo di servizio CVS predefinito `storageClass=software`, È necessario ottenere l'accesso alla funzionalità volumi sub-1TiB sul GCP per i numeri di progetto e gli ID progetto in questione. Ciò è necessario per Trident per eseguire il provisioning di volumi inferiori a 1 TiB. In caso contrario, le creazioni dei volumi **non avranno esito positivo** per i PVC con meno di 600 GiB. Ottenere l'accesso a volumi inferiori a 1 TiB utilizzando "questo modulo".

Il provisioning dei volumi creati da Trident per il livello di servizio CVS predefinito viene eseguito come segue:

- I PVC di dimensioni inferiori a 300 GiB creano un volume CVS da 300 GiB.
- I PVC compresi tra 300 GiB e 600 GiB determineranno la creazione di un volume CVS della dimensione richiesta da parte di Trident.
- I PVC compresi tra 600 GiB e 1 TIB determineranno la creazione di un volume CVS 1TiB da parte di Trident.
- I PVC che sono superiori a 1 TIB determineranno la creazione di un volume CVS della dimensione richiesta da parte di Trident.

Oltre ai valori di configurazione globali, quando si utilizza CVS su GCP, queste opzioni sono disponibili.

Opzione	Descrizione	Esempio
`apiRegion`	Regione dell'account CVS (obbligatoria). È la regione GCP in cui il backend eseguirà il provisioning dei volumi.	"us-west2"
`projectNumber`	Numero di progetto GCP (obbligatorio). Si trova nella schermata iniziale del portale Web GCP.	"123456789012"
`hostProjectNumber`	Numero di progetto host VPC condiviso GCP (richiesto se si utilizza un VPC condiviso)	"098765432109"
`apiKey`	Chiave API per l'account di servizio GCP con ruolo di amministratore CVS (obbligatorio). È il contenuto in formato JSON del file di chiave privata di un account di servizio GCP (copia integrale nel file di configurazione del backend). L'account del servizio deve avere il ruolo netappcloud.admin.	(contenuto del file delle chiavi private)
`secretKey`	Chiave segreta dell'account CVS (obbligatoria). Si trova nel portale web CVS in Impostazioni account > accesso API.	"predefinito"

Opzione	Descrizione	Esempio
`proxyURL`	URL proxy se il server proxy ha richiesto di connettersi all'account CVS. Il server proxy può essere un proxy HTTP o un proxy HTTPS. Nel caso di un proxy HTTPS, la convalida del certificato viene ignorata per consentire l'utilizzo di certificati autofirmati nel server proxy. **I server proxy con autenticazione abilitata non sono supportati**.	"http://proxy-server-hostname/"
`nfsMountOptions`	Opzioni di montaggio NFS; il valore predefinito è "-o nfsvers=3"	"nfsvers=3,proto=tcp,timeo=600"
`serviceLevel`	Livello di performance (standard, premium, Extreme), impostazione predefinita "standard"	"premium"
`network`	"Rete GCP utilizzata per i volumi CVS, impostazione predefinita "predefinita"	"predefinito"

Se si utilizza una rete VPC condivisa, è necessario specificare entrambi `projectNumber` e. `hostProjectNumber`. In tal caso, `projectNumber` è il progetto di servizio e. `hostProjectNumber` è il progetto host.

NetApp Cloud Volumes Service per GCP non supporta volumi CVS-Performance di dimensioni inferiori a 100 GiB o volumi CVS di dimensioni inferiori a 300 GiB. Per semplificare l'implementazione delle applicazioni, Trident crea automaticamente volumi di dimensioni minime se viene richiesto un volume troppo piccolo.

Quando si utilizza CVS su GCP, sono disponibili queste impostazioni predefinite delle opzioni del volume.

Opzione	Descrizione	Esempio
`exportRule`	Elenco di accesso NFS (indirizzi e/o subnet CIDR), valore predefinito "0.0.0.0/0"	"10.0.1.0/24,10.0.2.100"
`snapshotDir`	Controlla la visibilità di `.snapshot` directory	"falso"
`snapshotReserve`	Snapshot Reserve percent (percentuale riserva snapshot), il valore predefinito è "" per accettare il valore predefinito CVS pari a 0	"10"

Opzione	Descrizione	Esempio
`size`	Dimensione del volume, valore predefinito "100GiB"	"10T"

Esempio di CVS sul file di configurazione GCP

```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "apiRegion": "us-west2",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "1234567890123456789012345678901234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisI
sAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSa
PIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZN
chRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlz
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHi
sIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOgu
SaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyA
ZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz
lzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrt
HisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbO
guSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKe
yAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRA
GzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
```

```
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    },
    "proxyURL": "http://proxy-server-hostname/"
}
```

Configurazione di Azure NetApp Files

Per configurare e utilizzare un "Azure NetApp Files" back-end, è necessario quanto segue:

- `subscriptionID` Da un abbonamento Azure con Azure NetApp Files attivato
- `tenantID`, `clientID`, e. `clientSecret` da un "Registrazione dell'app" In Azure Active Directory con autorizzazioni sufficienti per il servizio Azure NetApp Files
- Ubicazione di Azure che ne contiene almeno una "subnet delegata"

Se si utilizza Azure NetApp Files per la prima volta o in una nuova posizione, è necessaria una configurazione iniziale di "guida rapida" ti guideranno.

Astra Trident 21.04.0 e versioni precedenti non supportano i pool di capacità QoS manuali.

Opzione	Descrizione	Predefinito
`version`	Sempre 1	
`storageDriverName`	"azure-netapp-files"	
`backendName`	Nome personalizzato per il backend dello storage	Nome del driver + "_" + caratteri casuali
`subscriptionID`	L'ID dell'abbonamento dell'abbonamento Azure	
`tenantID`	L'ID tenant di una registrazione app	
`clientID`	L'ID client di una registrazione dell'applicazione	
`clientSecret`	Il segreto del client da una registrazione dell'applicazione	
`serviceLevel`	Uno tra "Standard", "Premium" o "Ultra"	"" (casuale)

Opzione	Descrizione	Predefinito
`location`	Nome della posizione Azure in cui verranno creati nuovi volumi	"" (casuale)
`virtualNetwork`	Nome di una rete virtuale con una subnet delegata	"" (casuale)
`subnet`	Nome di una subnet delegata a. `Microsoft.Netapp/volumes`	"" (casuale)
`nfsMountOptions`	Controllo dettagliato delle opzioni di montaggio NFS	"-o nfsvers=3"
`limitVolumeSize`	Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore	"" (non applicato per impostazione predefinita)

> Il servizio Azure NetApp Files non supporta volumi di dimensioni inferiori a 100 GB. Per semplificare l'implementazione delle applicazioni, Trident crea automaticamente volumi da 100 GB se viene richiesto un volume più piccolo.

Per impostazione predefinita, è possibile controllare il provisioning di ciascun volume utilizzando queste opzioni in una sezione speciale della configurazione.

Opzione	Descrizione	Predefinito
`exportRule`	Regola o regole di esportazione per i nuovi volumi. Deve essere un elenco separato da virgole di qualsiasi combinazione di indirizzi IPv4 o subnet IPv4 nella notazione CIDR.	"0.0.0.0/0"
`snapshotDir`	Controlla la visibilità di `.snapshot` directory	"falso"
`size`	La dimensione predefinita dei nuovi volumi	"100 G"

Configurazioni Azure NetApp Files di esempio

Esempio 1: Configurazione backend minima per Azure-netapp-Files

Questa è la configurazione backend minima assoluta. Con questa configurazione, Trident scoprirà tutti gli account NetApp, i pool di capacità e le subnet delegate ad ANF in ogni sede in tutto il mondo e inserirà nuovi volumi in uno di essi in maniera casuale.

Questa configurazione è utile quando si inizia a utilizzare ANF e si provano le cose, tuttavia, in pratica, è necessario fornire un ambito aggiuntivo per i volumi che si effettua il provisioning per assicurarsi che abbiano

le caratteristiche desiderate e finiscano in una rete vicina al calcolo che lo utilizza. Per ulteriori dettagli, vedere gli esempi successivi.

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
    "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
    "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
    "clientSecret": "SECRET"
}
```

Esempio 2: Singola posizione e livello di servizio specifico per Azure-netapp-Files

Questa configurazione di back-end colloca i volumi nella posizione "eastus" di Azure in un pool di capacità "Premium". Trident rileva automaticamente tutte le sottoreti delegate ad ANF in quella posizione e inserisce un nuovo volume su una di esse in modo casuale.

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
    "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
    "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
    "clientSecret": "SECRET",
    "location": "eastus",
    "serviceLevel": "Premium"
}
```

Esempio 3: Configurazione avanzata per Azure-netapp-Files

Questa configurazione di back-end riduce ulteriormente l’ambito del posizionamento del volume in una singola subnet e modifica alcune impostazioni predefinite di provisioning del volume.

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
    "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
    "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
    "clientSecret": "SECRET",
    "location": "eastus",
    "serviceLevel": "Premium",
    "virtualNetwork": "my-virtual-network",
    "subnet": "my-subnet",
    "nfsMountOptions": "nfsvers=3,proto=tcp,timeo=600",
    "limitVolumeSize": "500Gi",
    "defaults": {
        "exportRule": "10.0.0.0/24,10.0.1.0/24,10.0.2.100",
        "size": "200Gi"
    }
}
```

Esempio 4: Pool di storage virtuali con file Azure-netapp

Questa configurazione di back-end definisce più configurazioni "pool di storage" in un singolo file. Ciò è utile quando si dispone di più pool di capacità che supportano diversi livelli di servizio e si desidera creare classi di storage in Kubernetes che ne rappresentano.

Questo sta semplicemente graffiando la superficie della potenza dei pool di storage virtuali e delle relative etichette.

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
    "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
    "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
    "clientSecret": "SECRET",
    "nfsMountOptions": "nfsvers=3,proto=tcp,timeo=600",
    "labels": {
        "cloud": "azure"
    },
    "location": "eastus",

    "storage": [
        {
            "labels": {
                "performance": "gold"
            },
            "serviceLevel": "Ultra"
        },
        {
            "labels": {
                "performance": "silver"
            },
            "serviceLevel": "Premium"
        },
        {
            "labels": {
                "performance": "bronze"
            },
            "serviceLevel": "Standard",
        }
    ]
}
```

Problemi noti e limitazioni

Informazioni su problemi e limitazioni noti durante l'utilizzo di Astra Trident con Docker.

L'aggiornamento del plug-in Trident Docker Volume alla versione 20.10 e successive da versioni precedenti comporta un errore di aggiornamento con l'errore NO tali file o directory.

Soluzione alternativa

1. Disattivare il plug-in.

```
docker plugin disable -f netapp:latest
```

2. Rimuovere il plug-in.

```
docker plugin rm -f netapp:latest
```

3. Reinstallare il plug-in fornendo il plug-in extra `config` parametro.

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant
-all-permissions config=config.json
```

I nomi dei volumi devono contenere almeno 2 caratteri.

> Si tratta di una limitazione del client Docker. Il client interpreta un nome di singolo carattere come percorso Windows. "Vedere il bug 25773".

Docker Swarm ha alcuni comportamenti che impediscono ad Astra Trident di supportarlo con ogni combinazione di storage e driver.

- Docker Swarm utilizza attualmente il nome del volume anziché l'ID del volume come identificatore univoco del volume.
- Le richieste di volume vengono inviate simultaneamente a ciascun nodo di un cluster Swarm.
- I plug-in dei volumi (incluso Astra Trident) devono essere eseguiti in modo indipendente su ciascun nodo di un cluster Swarm. Grazie al funzionamento di ONTAP e al modo in cui funziona `ontap-nas` e. `ontap-san` i driver funzionano, sono gli unici ad essere in grado di funzionare entro questi limiti.

Il resto dei piloti è soggetto a problemi come le condizioni di gara che possono comportare la creazione di un gran numero di volumi per una singola richiesta senza un "vincitore" chiaro; ad esempio, Element ha una funzione che consente ai volumi di avere lo stesso nome ma ID diversi.

NetApp ha fornito feedback al team Docker, ma non ha alcuna indicazione di ricorso futuro.

Se viene eseguito il provisioning di un FlexGroup, ONTAP non esegue il provisioning di un secondo FlexGroup se il secondo FlexGroup ha uno o più aggregati in comune con il FlexGroup sottoposto a provisioning.

Domande frequenti

Trova le risposte alle domande frequenti sull'installazione, la configurazione, l'aggiornamento e la risoluzione dei problemi di Astra Trident.

Domande generali

Con quale frequenza viene rilasciato Astra Trident?

Astra Trident viene rilasciato ogni tre mesi: Gennaio, aprile, luglio e ottobre. Questo è un mese dopo il rilascio di Kubernetes.

Astra Trident supporta tutte le funzionalità rilasciate in una determinata versione di Kubernetes?

Astra Trident di solito non supporta le funzionalità alpha in Kubernetes. Trident potrebbe supportare le funzionalità beta all'interno delle due release Trident che seguono la release beta di Kubernetes.

Astra Trident dipende dal funzionamento di altri prodotti NetApp?

Astra Trident non ha dipendenze da altri prodotti software NetApp e funziona come applicazione standalone. Tuttavia, è necessario disporre di un dispositivo di storage back-end NetApp.

Come posso ottenere i dettagli completi della configurazione di Astra Trident?

Utilizzare `tridentctl get` Per ottenere ulteriori informazioni sulla configurazione di Astra Trident.

Posso ottenere metriche su come viene eseguito il provisioning dello storage da Astra Trident?

Sì. Trident 20.01 introduce gli endpoint Prometheus che possono essere utilizzati per raccogliere informazioni sul funzionamento di Astra Trident, come il numero di backend gestiti, il numero di volumi sottoposti a provisioning, i byte utilizzati e così via. È inoltre possibile utilizzare Cloud Insights per il monitoraggio e l'analisi.

L'esperienza dell'utente cambia quando si utilizza Astra Trident come provider CSI?

No Non ci sono modifiche per quanto riguarda l'esperienza e le funzionalità dell'utente. Il nome del provider utilizzato è `csi.trident.netapp.io`. Questo metodo di installazione di Astra Trident è consigliato se si desidera utilizzare tutte le nuove funzionalità fornite dalle release attuali e future.

Installare e utilizzare Astra Trident su un cluster Kubernetes

Quali sono le versioni supportate di `etcd`?

Astra Trident non ha più bisogno di un `etcd`. Utilizza i CRD per mantenere lo stato.

Astra Trident supporta un'installazione offline da un registro privato?

Sì, Astra Trident può essere installato offline. Vedere "qui".

Posso installare Astra Trident in remoto?

Sì. Astra Trident 18.10 e versioni successive supportano la funzionalità di installazione remota da qualsiasi computer `kubectl` accesso al cluster. Dopo `kubectl` l'accesso viene verificato (ad esempio, avviare un `kubectl get nodes` dal computer remoto per verificare), seguire le istruzioni di installazione.

Posso configurare l'alta disponibilità con Astra Trident?

Astra Trident viene installato come Kubernetes Deployment (ReplicaSet) con un'istanza e quindi ha integrato. Non aumentare il numero di repliche nella distribuzione. Se il nodo in cui è installato Astra Trident viene perso o il pod è altrimenti inaccessibile, Kubernetes ridistribuisce automaticamente il pod su un nodo integro nel cluster. Astra Trident è solo il piano di controllo, pertanto i pod attualmente montati non vengono influenzati se Astra Trident viene riimplementato.

Astra Trident ha bisogno di accedere allo spazio dei nomi del sistema kube?

Astra Trident legge dal Kubernetes API Server per determinare quando le applicazioni richiedono nuovi PVC, in modo da avere accesso al sistema kube.

Quali sono i ruoli e i privilegi utilizzati da Astra Trident?

Il programma di installazione di Trident crea Kubernetes ClusterRole, che ha accesso specifico alle risorse PersistentVolume, PersistentVolumeClaim, StorageClass e Secret del cluster Kubernetes. Vedere "qui".

È possibile generare localmente i file manifest utilizzati da Astra Trident per l'installazione?

È possibile generare e modificare localmente i file manifest utilizzati da Astra Trident per l'installazione, se necessario. Vedere "qui".

Posso condividere la stessa SVM backend ONTAP per due istanze separate di Astra Trident per due cluster Kubernetes separati?

Sebbene non sia consigliato, è possibile utilizzare lo stesso SVM backend per due istanze di Astra Trident. Specificare un nome di volume univoco per ogni istanza durante l'installazione e/o specificare un nome univoco `StoragePrefix` nel `setup/backend.json` file. In questo modo si garantisce che non venga utilizzato lo stesso FlexVol per entrambe le istanze.

È possibile installare Astra Trident sotto ContainerLinux (in precedenza CoreOS)?

Astra Trident è semplicemente un pod Kubernetes e può essere installato ovunque sia in esecuzione Kubernetes.

Posso utilizzare Astra Trident con NetApp Cloud Volumes ONTAP?

Sì, Astra Trident è supportato su AWS, Google Cloud e Azure.

Astra Trident funziona con Cloud Volumes Services?

Sì, Astra Trident supporta il servizio Azure NetApp Files in Azure e Cloud Volumes Service in GCP.

Risoluzione dei problemi e supporto

NetApp supporta Astra Trident?

Anche se Astra Trident è open source e viene fornito gratuitamente, NetApp lo supporta completamente, a condizione che il vostro back-end NetApp sia supportato.

Come si fa a inoltrare un caso di supporto?

Per inoltrare un caso di supporto, eseguire una delle seguenti operazioni:

1. Contatta il tuo Support account Manager e ricevi assistenza per la richiesta di un ticket.
2. Inoltrare un caso di supporto contattando "Supporto NetApp".

Come si genera un bundle di log di supporto?

È possibile creare un bundle di supporto eseguendo `tridentctl logs -a`. Oltre ai log acquisiti nel bundle, acquisire il log del kubelet per diagnosticare i problemi di montaggio sul lato Kubernetes. Le istruzioni per ottenere il log di Kubernetes variano in base alla modalità di installazione di Kubernetes.

Cosa devo fare se devo inoltrare una richiesta per una nuova funzionalità?

Creare un problema su "Trident Github" E menziona **RFE** nell'oggetto e nella descrizione del problema.

Dove posso segnalare un difetto?

Creare un problema su "Astra Trident Github". Assicurarsi di includere tutte le informazioni e i registri necessari relativi al problema.

Cosa succede se ho domande rapide su Astra Trident su cui ho bisogno di chiarimenti? Esiste una community o un forum?

In caso di domande, problemi o richieste, contattaci tramite il nostro "Lasco" Team o GitHub.

La password del sistema di storage è cambiata e Astra Trident non funziona più, come posso ripristinarla?

Aggiornare la password del backend con `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`. Sostituire `myBackend` nell'esempio con il nome backend, e. `/path/to_new_backend.json` con il percorso verso il corretto `backend.json` file.

Astra Trident non riesce a trovare il nodo Kubernetes. Come posso risolvere questo problema?

Ci sono due scenari probabili per cui Astra Trident non riesce a trovare un nodo Kubernetes. Può essere dovuto a un problema di rete all'interno di Kubernetes o a un problema DNS. Il demonset di nodi Trident eseguito su ciascun nodo Kubernetes deve essere in grado di comunicare con il controller Trident per

registrare il nodo con Trident. Se si verificano modifiche di rete dopo l'installazione di Astra Trident, il problema si verifica solo con i nuovi nodi Kubernetes aggiunti al cluster.

Se il pod Trident viene distrutto, perderò i dati?

I dati non andranno persi se il pod Trident viene distrutto. I metadati di Trident vengono memorizzati negli oggetti CRD. Tutti i PVS forniti da Trident funzioneranno normalmente.

Aggiorna Astra Trident

È possibile eseguire l'aggiornamento da una versione precedente direttamente a una versione più recente (ignorando alcune versioni)?

NetApp supporta l'aggiornamento di Astra Trident da una release principale alla successiva release principale immediata. È possibile eseguire l'aggiornamento dalla versione 18.xx alla versione 19.xx, dalla versione 19.xx alla versione 20.xx e così via. Prima dell'implementazione in produzione, è necessario testare l'aggiornamento in un laboratorio.

È possibile eseguire il downgrade di Trident a una release precedente?

Se si desidera eseguire il downgrade, è necessario valutare diversi fattori. Vedere "la sezione sul downgrade".

Gestione di back-end e volumi

È necessario definire le LIF di gestione e dati in un file di definizione back-end ONTAP?

NetApp consiglia di inserire entrambe le opzioni nel file di definizione del backend. Tuttavia, la LIF di gestione è l'unica a essere obbligatoria.

Astra Trident può configurare CHAP per i backend ONTAP?

Sì. A partire da 20.04, Astra Trident supporta CHAP bidirezionale per backend ONTAP. Questa operazione richiede l'impostazione `useCHAP=true` nella configurazione back-end.

Come posso gestire le policy di esportazione con Astra Trident?

Astra Trident è in grado di creare e gestire dinamicamente le policy di esportazione a partire dalla versione 20.04. Ciò consente all'amministratore dello storage di fornire uno o più blocchi CIDR nella configurazione di back-end e di aggiungere IP di nodo che rientrano in questi intervalli a un criterio di esportazione creato da Trident. In questo modo, Astra Trident gestisce automaticamente l'aggiunta e l'eliminazione di regole per i nodi con IP all'interno di dati CIDR. Questa funzione richiede CSI Trident.

È possibile specificare una porta in DataLIF?

Astra Trident 19.01 e versioni successive supportano la specifica di una porta nel DataLIF. Configurarlo in `backend.json` file as (file con nome "`managementLIF": <ip address>:<port>`". Ad esempio, se l'indirizzo IP della LIF di gestione è 192.0.2.1 e la porta è 1000, configurare `"managementLIF": "192.0.2.1:1000"`.

È possibile utilizzare gli indirizzi IPv6 per le LIF di gestione e dati?

Sì. Astra Trident 20.01 supporta la definizione degli indirizzi IPv6 per i parametri di gestione LIF e dataLIF per i backend ONTAP. Assicurarsi che l'indirizzo segua la semantica IPv6 e che la managementLIF sia definita tra parentesi quadre (ad esempio, `[ec0d:6504:a9c1:ae67:53d1:4bdf:ab32:e233]`). Assicurarsi inoltre che Astra Trident sia installato utilizzando `` `--use-ipv6 `` Flag per il funzionamento su IPv6.

È possibile aggiornare la LIF di gestione sul back-end?

Sì, è possibile aggiornare la LIF di gestione back-end utilizzando `tridentctl update backend` comando.

È possibile aggiornare Data LIF sul back-end?

No, non è possibile aggiornare Data LIF sul back-end.

Posso creare backend multipli in Astra Trident per Kubernetes?

Astra Trident supporta molti backend contemporaneamente, con lo stesso driver o driver diversi.

In che modo Astra Trident memorizza le credenziali di back-end?

Astra Trident memorizza le credenziali di back-end come Kubernetes Secrets.

In che modo Astra Trident seleziona un backend specifico?

Se non è possibile utilizzare gli attributi di backend per selezionare automaticamente i pool giusti per una classe, il `storagePools` e. `additionalStoragePools` i parametri vengono utilizzati per selezionare un set specifico di pool.

Come posso garantire che Astra Trident non effettuerà il provisioning da un backend specifico?

Il `excludeStoragePools` Il parametro viene utilizzato per filtrare il set di pool che Astra Trident utilizzerà per il provisioning e rimuoverà i pool corrispondenti.

Se esistono più backend dello stesso tipo, come fa Astra Trident a selezionare quale backend utilizzare?

Se sono presenti più backend configurati dello stesso tipo, Astra Trident seleziona il backend appropriato in base ai parametri presenti in `StorageClass` e. `PersistentVolumeClaim`. Ad esempio, se sono presenti più backend di driver ontap-nas, Astra Trident tenta di associare i parametri in `StorageClass` e. `PersistentVolumeClaim` combinato e abbinato a un backend in grado di soddisfare i requisiti elencati nella `StorageClass` e. `PersistentVolumeClaim`. Se ci sono più backend che corrispondono alla richiesta, Astra Trident seleziona uno di essi in maniera casuale.

Astra Trident supporta CHAP bidirezionale con Element/SolidFire?

Sì.

In che modo Astra Trident implementa Qtree su un volume ONTAP? Quanti Qtree possono essere implementati su un singolo volume?

Il `ontap-nas-economy` Il driver crea fino a 200 Qtree nello stesso FlexVol (configurabile tra 50 e 300), 100,000 Qtree per nodo cluster e 2,4 milioni per cluster. Quando si immette un nuovo `PersistentVolumeClaim` Che è servito dal driver economico, il driver cerca di vedere se esiste già un FlexVol in grado di servire il nuovo Qtree. Se il FlexVol non esiste in grado di servire il Qtree, viene creato un nuovo FlexVol.

Come si impostano le autorizzazioni Unix per i volumi forniti su NAS ONTAP?

È possibile impostare i permessi Unix sul volume fornito da Astra Trident impostando un parametro nel file di definizione del backend.

Come posso configurare un set esplicito di opzioni di montaggio NFS di ONTAP durante il provisioning di un volume?

Per impostazione predefinita, Astra Trident non imposta le opzioni di montaggio su alcun valore con Kubernetes. Per specificare le opzioni di montaggio nella classe di storage Kubernetes, seguire l'esempio fornito "qui".

Come si impostano i volumi sottoposti a provisioning in base a una policy di esportazione specifica?

Per consentire agli host appropriati di accedere a un volume, utilizzare `exportPolicy` parametro configurato nel file di definizione del backend.

Come si imposta la crittografia del volume tramite Astra Trident con ONTAP?

È possibile impostare la crittografia sul volume fornito da Trident utilizzando il parametro di crittografia nel file di definizione del backend.

Qual è il modo migliore per implementare la QoS per ONTAP attraverso Astra Trident?

Utilizzare `StorageClasses` Per implementare QoS per ONTAP.

Come si specifica il thin provisioning o thick provisioning tramite Astra Trident?

I driver ONTAP supportano il thin provisioning o il thick provisioning. Per impostazione predefinita, i driver ONTAP passano al thin provisioning. Se si desidera eseguire il thick provisioning, è necessario configurare il file di definizione del backend o il `StorageClass`. Se entrambi sono configurati, `StorageClass` ha la precedenza. Configurare quanto segue per ONTAP:

1. Acceso `StorageClass`, impostare `provisioningType` attributo come thick.
2. Nel file di definizione del backend, attivare i volumi thick impostando `backend spaceReserve parameter` come volume.

Come si può verificare che i volumi utilizzati non vengano cancellati anche se si elimina accidentalmente il PVC?

La protezione PVC viene attivata automaticamente su Kubernetes a partire dalla versione 1.10.

Posso far crescere i PVC NFS creati da Astra Trident?

Sì. È possibile espandere un PVC creato da Astra Trident. Tenere presente che la crescita automatica del volume è una funzione di ONTAP non applicabile a Trident.

Se si dispone di un volume creato all'esterno di Astra Trident, è possibile importarlo in Astra Trident?

A partire dalla versione 19.04, è possibile utilizzare la funzione di importazione dei volumi per trasferire i volumi in Kubernetes.

È possibile importare un volume in modalità SnapMirror Data Protection (DP) o offline?

L'importazione del volume non riesce se il volume esterno è in modalità DP o non è in linea. Viene visualizzato il seguente messaggio di errore:

```
Error: could not import volume: volume import failed to get size of
volume: volume <name> was not found (400 Bad Request) command terminated
with exit code 1.
Make sure to remove the DP mode or put the volume online before importing
the volume.
```

È possibile espandere i PVC iSCSI creati da Astra Trident?

Trident 19.10 supporta l'espansione del PVS iSCSI utilizzando CSI Provisioner.

Come viene tradotta la quota di risorse in un cluster NetApp?

La quota delle risorse di storage di Kubernetes dovrebbe funzionare finché lo storage NetApp dispone di capacità. Quando lo storage NetApp non riesce a rispettare le impostazioni di quota di Kubernetes a causa della mancanza di capacità, Astra Trident tenta di eseguire il provisioning, ma gli errori non vengono eseguiti.

È possibile creare snapshot di volumi utilizzando Astra Trident?

Sì. Astra Trident supporta la creazione di snapshot di volumi on-demand e volumi persistenti. Per creare PVS da snapshot, assicurarsi che `VolumeSnapshotDataSource` feature gate è stato attivato.

Quali sono i driver che supportano le snapshot dei volumi Astra Trident?

A partire da oggi, il supporto on-demand per le snapshot è disponibile per il nostro `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, e. `azure-netapp-files` driver di back-end.

Come si esegue un backup snapshot di un volume fornito da Astra Trident con ONTAP?

Disponibile in `ontap-nas`, `ontap-san`, e. `ontap-nas-flexgroup` driver. È inoltre possibile specificare un `snapshotPolicy` per `ontap-san-economy` Driver a livello di FlexVol.

Questa opzione è disponibile anche sul `ontap-nas-economy` Ma a livello di granularità FlexVol e non a livello di granularità qtree. Per abilitare la funzione di snapshot dei volumi forniti da Astra Trident, impostare l'opzione del parametro backend `snapshotPolicy` Al criterio di snapshot desiderato, come definito nel backend ONTAP. Astra Trident non conosce le snapshot eseguite dal controller di storage.

È possibile impostare una percentuale di riserva di snapshot per un volume fornito tramite Astra Trident?

Sì, è possibile riservare una percentuale specifica di spazio su disco per la memorizzazione delle copie Snapshot tramite Astra Trident impostando `snapshotReserve` nel file di definizione del backend. Se è stato configurato `snapshotPolicy` e. `snapshotReserve` nel file di definizione del backend, la percentuale di riserva snapshot viene impostata in base a. `snapshotReserve` percentuale indicata nel file backend. Se il `snapshotReserve` Il numero percentuale non viene menzionato, ONTAP per impostazione predefinita assume la percentuale di riserva snapshot come 5. Se il `snapshotPolicy` l'opzione è impostata su nessuno, la percentuale di riserva snapshot è impostata su 0.

È possibile accedere direttamente alla directory di snapshot del volume e copiare i file?

Sì, è possibile accedere alla directory di snapshot sul volume fornito da Trident impostando `snapshotDir` nel file di definizione back-end.

È possibile configurare SnapMirror per i volumi tramite Astra Trident?

Attualmente, SnapMirror deve essere impostato esternamente utilizzando l'interfaccia CLI di ONTAP o Gestione di sistema di OnCommand.

Come si ripristinano i volumi persistenti in uno snapshot ONTAP specifico?

Per ripristinare un volume in uno snapshot ONTAP, attenersi alla seguente procedura:

1. Interrompere il pod dell'applicazione che utilizza il volume persistente.
2. Ripristinare lo snapshot richiesto tramite l'interfaccia utente di ONTAP o Gestione di sistema di OnCommand.
3. Riavviare il pod applicazioni.

Trident può eseguire il provisioning di volumi su SVM con un mirror di condivisione del carico configurato?

È possibile creare mirror di condivisione del carico per i volumi root delle SVM che servono dati su NFS. ONTAP aggiorna automaticamente i mirror di condivisione del carico per i volumi creati da Trident. Ciò potrebbe causare ritardi nell'installazione dei volumi. Quando si creano più volumi utilizzando Trident, il provisioning di un volume dipende dall'aggiornamento del mirror di condivisione del carico da parte di ONTAP.

Come è possibile separare l’utilizzo della classe di storage per ciascun cliente/tenant?

Kubernetes non consente classi di storage negli spazi dei nomi. Tuttavia, è possibile utilizzare Kubernetes per limitare l’utilizzo di una classe di storage specifica per spazio dei nomi utilizzando le quote delle risorse di storage, che sono per spazio dei nomi. Per negare l’accesso a uno spazio dei nomi specifico a uno storage specifico, impostare la quota di risorse su 0 per tale classe di storage.

Supporto

Astra Trident è un progetto NetApp ufficialmente supportato. Puoi contattare NetApp utilizzando uno qualsiasi dei meccanismi standard e ottenere il supporto di livello Enterprise di cui hai bisogno.

Sul è presente anche una vivace community pubblica di utenti di container (inclusi gli sviluppatori Astra Trident) `containers` canale attivato "Il lavoro di NetApp". Questo è un ottimo posto per porre domande generali sul progetto e discutere argomenti correlati con colleghi che condividono la stessa opinione.

Risoluzione dei problemi

Utilizzare i puntatori forniti qui per la risoluzione dei problemi che potrebbero verificarsi durante l'installazione e l'utilizzo di Astra Trident.

> Per assistenza con Astra Trident, creare un pacchetto di supporto utilizzando `tridentctl logs -a -n trident` e inviarla a. `NetApp Support <Getting Help>`.

> Per un elenco completo degli articoli per la risoluzione dei problemi, consultare "Knowledge base di NetApp (accesso richiesto)". È inoltre possibile trovare informazioni sulla risoluzione dei problemi relativi ad Astra "qui".

Risoluzione dei problemi generali

- Se il pod Trident non si accende correttamente (ad esempio, quando il pod Trident è bloccato in `ContainerCreating` con meno di due container pronti), in esecuzione `kubectl -n trident describe deployment trident` e. `kubectl -n trident describe pod trident--**` può fornire ulteriori informazioni. Ottenere i log di kubelet (ad esempio, via `journalctl -xeu kubelet`) può anche essere utile.
- Se i log di Trident non contengono informazioni sufficienti, provare ad attivare la modalità di debug per Trident passando il `-d` contrassegnare il parametro install in base all'opzione di installazione.

  Quindi confermare che il debug sia impostato utilizzando `./tridentctl logs -n trident` e alla ricerca `level=debug msg` nel log.

  Installato con l'operatore

  ```
  # kubectl patch torc trident -n <namespace> --type=merge -p
  '{"spec":{"debug":true}}'
  ```

  In questo modo verranno riavviati tutti i pod Trident, che possono richiedere alcuni secondi. È possibile verificare questa condizione osservando la colonna 'ETÀ' nell'output di `kubectl get pod -n trident`.

  Per l'utilizzo di Astra Trident 20.07 e 20.10 `tprov` al posto di `torc`.

  Installato con Helm

  ```
  $ helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
  tridentDebug=true`
  ```

  Installato con tridentctl

  ```
  ./tridentctl uninstall -n trident
  ./tridentctl install -d -n trident
  ```

- È inoltre possibile ottenere i log di debug per ciascun backend includendo `debugTraceFlags` nella

definizione di back-end. Ad esempio, Includi `debugTraceFlags: {"api":true, "method":true,}` Per ottenere chiamate API e attraversamenti dei metodi nei log di Trident. I backend esistenti possono avere `debugTraceFlags` configurato con un `tridentctl backend update`.

- Quando si utilizza RedHat CoreOS, assicurarsi che `iscsid` è attivato sui nodi di lavoro e avviato per impostazione predefinita. Questa operazione può essere eseguita utilizzando OpenShift MachineConfigs o modificando i modelli di accensione.
- Si tratta di un problema comune che potrebbe verificarsi quando si utilizza Trident con "Azure NetApp Files" è quando i segreti del tenant e del client provengono da una registrazione dell'applicazione con autorizzazioni insufficienti. Per un elenco completo dei requisiti Trident, vedere "Azure NetApp Files" configurazione.
- In caso di problemi con il montaggio di un PV su un container, assicurarsi che `rpcbind` è installato e in esecuzione. Utilizzare il gestore dei pacchetti richiesto per il sistema operativo host e verificare se `rpcbind` è in esecuzione. È possibile controllare lo stato di `rpcbind` eseguire un `systemctl status rpcbind` o equivalente.
- Se un backend Trident segnala che si trova in `failed` stato nonostante abbia lavorato in precedenza, è probabile che sia causato dalla modifica delle credenziali SVM/admin associate al backend. Aggiornamento delle informazioni di back-end tramite `tridentctl update backend` O rimbalzare il pod Trident risolverà questo problema.
- Se stai aggiornando il tuo cluster Kubernetes e/o Trident per utilizzare le snapshot dei volumi beta, assicurati che tutti i CRS di snapshot alfa esistenti siano completamente rimossi. È quindi possibile utilizzare `tridentctl obliviate alpha-snapshot-crd` Comando per eliminare i CRD snapshot alfa. Vedere "questo blog" comprendere i passaggi necessari per la migrazione delle snapshot alfa.
- Se si riscontrano problemi di autorizzazione durante l'installazione di Trident con Docker come runtime del container, tentare l'installazione di Trident con `--in cluster=false` allarme. Questo non utilizzerà un pod di installazione ed eviterà i problemi di autorizzazione causati da `trident-installer` utente.
- Utilizzare `uninstall parameter <Uninstalling Trident>` per la pulizia dopo un'esecuzione non riuscita. Per impostazione predefinita, lo script non rimuove i CRD creati da Trident, rendendo sicuro disinstallare e installare di nuovo anche in una distribuzione in esecuzione.
- Se si desidera eseguire il downgrade a una versione precedente di Trident, eseguire prima il `tridenctl uninstall` Comando per rimuovere Trident. Scaricare il desiderato "Versione di Trident" e installare utilizzando `tridentctl install` comando. Considerare un downgrade solo se non sono stati creati nuovi PVS e se non sono state apportate modifiche alle classi di storage/backend/PVS già esistenti. Poiché Trident utilizza ora i CRD per mantenere lo stato, tutte le entità di storage create (backend, classi di storage, PVS e snapshot dei volumi) ne hanno `associated CRD objects <Kubernetes CustomResourceDefinition Objects>` Invece dei dati scritti nel PV utilizzati dalla versione precedente di Trident. **Il PVS appena creato non sarà utilizzabile quando si torna a una versione precedente. le modifiche apportate agli oggetti, come backend, PVS, classi di storage e snapshot di volumi (creati/aggiornati/cancellati) non saranno visibili a Trident quando si esegue il downgrade**. Il PV utilizzato dalla versione precedente di Trident installata sarà ancora visibile a Trident. Il ritorno a una versione precedente non interrompe l'accesso ai PVS già creati utilizzando la versione precedente, a meno che non siano stati aggiornati.
- Per rimuovere completamente Trident, eseguire `tridentctl obliviate crd` comando. In questo modo verranno rimossi tutti gli oggetti CRD e i CRD non saranno definiti. Trident non gestirà più i PVS già forniti.

  Trident dovrà essere riconfigurato da zero.

- Una volta completata correttamente l'installazione, se un PVC è bloccato in `Pending` fase, esecuzione

`kubectl describe pvc` Può fornire ulteriori informazioni sul motivo per cui Trident non ha eseguito il provisioning di un PV per questo PVC.

Risoluzione dei problemi di un'implementazione Trident non riuscita utilizzando l'operatore

Se si sta implementando Trident utilizzando l'operatore, lo stato di `TridentOrchestrator` modifiche da `Installing` a. `Installed`. Se si osserva `Failed` e l'operatore non è in grado di eseguire il ripristino da solo, controllare i log dell'operatore eseguendo il seguente comando:

```
tridentctl logs -l trident-operator
```

L'uscita dei log del container trident-operator può indicare dove si trova il problema. Ad esempio, uno di questi problemi potrebbe essere l'impossibilità di estrarre le immagini container richieste dai registri upstream in un ambiente Airgapped.

Per capire perché l'installazione di Trident non è riuscita, consultare `TridentOrchestrator` stato.

```
$ kubectl describe torc trident-2
Name:         trident-2
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Status:
  Current Installation Params:
    IPv6:
    Autosupport Hostname:
    Autosupport Image:
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:
    Enable Node Prep:
    Image Pull Secrets:         <nil>
    Image Registry:
    k8sTimeout:
    Kubelet Dir:
    Log Format:
    Silence Autosupport:
    Trident Image:
  Message:                      Trident is bound to another CR 'trident'
  Namespace:                    trident-2
  Status:                       Error
  Version:
Events:
  Type     Reason  Age                From                        Message
  ----     ------  ----               ----                        -------
  Warning  Error   16s (x2 over 16s)  trident-operator.netapp.io  Trident
is bound to another CR 'trident'
```

Questo errore indica che esiste già un `TridentOrchestrator`Utilizzato per installare Trident. Poiché ogni cluster Kubernetes può avere una sola istanza di Trident, l'operatore garantisce che in qualsiasi momento esista una sola istanza attiva `TridentOrchestrator` che può creare.

Inoltre, osservare lo stato dei pod Trident può spesso indicare se qualcosa non è giusto.

```
$ kubectl get pods -n trident

NAME                                READY   STATUS             RESTARTS
AGE
trident-csi-4p5kq                   1/2     ImagePullBackOff   0
5m18s
trident-csi-6f45bfd8b6-vfrkw        4/5     ImagePullBackOff   0
5m19s
trident-csi-9q5xc                   1/2     ImagePullBackOff   0
5m18s
trident-csi-9v95z                   1/2     ImagePullBackOff   0
5m18s
trident-operator-766f7b8658-ldzsv   1/1     Running            0
8m17s
```

È possibile notare che i pod non sono in grado di inizializzare completamente perché una o più immagini container non sono state recuperate.

Per risolvere il problema, modificare `TridentOrchestrator` CR. In alternativa, è possibile eliminare `TridentOrchestrator`e crearne uno nuovo con la definizione modificata e precisa.

Risoluzione dei problemi di un'implementazione Trident non riuscita utilizzando `tridentctl`

Per capire cosa è andato storto, è possibile eseguire di nuovo il programma di installazione utilizzando `-d` argomento, che attiverà la modalità di debug e ti aiuterà a capire qual è il problema:

```
./tridentctl install -n trident -d
```

Dopo aver risolto il problema, è possibile eseguire l'installazione come segue, quindi eseguire `tridentctl install` di nuovo comando:

```
./tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Removed Trident user from security context constraint.
INFO Trident uninstallation succeeded.
```

Best practice e consigli

Implementazione

Quando si implementa Astra Trident, utilizzare i consigli elencati di seguito.

Eseguire l'implementazione in uno spazio dei nomi dedicato

"Spazi dei nomi" fornire una separazione amministrativa tra diverse applicazioni e costituire una barriera per la condivisione delle risorse. Ad esempio, un PVC di uno spazio dei nomi non può essere utilizzato da un altro. Astra Trident fornisce risorse PV a tutti gli spazi dei nomi nel cluster Kubernetes e sfrutta di conseguenza un account di servizio con privilegi elevati.

Inoltre, l'accesso al pod Trident potrebbe consentire a un utente di accedere alle credenziali del sistema di storage e ad altre informazioni sensibili. È importante assicurarsi che gli utenti delle applicazioni e le applicazioni di gestione non abbiano la possibilità di accedere alle definizioni degli oggetti Trident o ai pod stessi.

Utilizza quote e limiti di intervallo per controllare il consumo dello storage

Kubernetes dispone di due funzionalità che, se combinate, offrono un potente meccanismo per limitare il consumo di risorse da parte delle applicazioni. Il "meccanismo di quota dello storage" consente all'amministratore di implementare limiti di consumo di capacità e numero di oggetti globali e specifici per classe di storage in base allo spazio dei nomi. Inoltre, utilizzando un "limite di intervallo" Garantisce che le richieste PVC rientrino in un valore minimo e massimo prima che la richiesta venga inoltrata al provisioning.

Questi valori sono definiti in base allo spazio dei nomi, il che significa che ogni spazio dei nomi deve avere valori definiti che sono in linea con i requisiti delle risorse. Vedere qui per informazioni su "come sfruttare le quote".

Configurazione dello storage

Ogni piattaforma di storage del portfolio NetApp dispone di funzionalità uniche che offrono vantaggi alle applicazioni, containerizzate o meno. Trident funziona con ONTAP ed Element. Non esiste una piattaforma più adatta a tutte le applicazioni e gli scenari rispetto all'altra, tuttavia, è necessario tenere conto delle esigenze dell'applicazione e del team che amministra il dispositivo quando si sceglie una piattaforma.

Seguire le Best practice di base per il sistema operativo host con il protocollo che si sta sfruttando. Se lo si desidera, si consiglia di includere Best practice applicative, se disponibili, con impostazioni di backend, classe di storage e PVC per ottimizzare lo storage per applicazioni specifiche.

Best practice per ONTAP e Cloud Volumes ONTAP

Scopri le Best practice per la configurazione di ONTAP e Cloud Volumes ONTAP per Trident.

I seguenti consigli sono linee guida per la configurazione di ONTAP per i carichi di lavoro containerizzati, che consumano volumi che vengono forniti dinamicamente da Trident. Ciascuno di essi deve essere considerato e valutato per l'adeguatezza nel proprio ambiente.

Utilizzare SVM dedicate a Trident

Le macchine virtuali di storage (SVM) forniscono isolamento e separazione amministrativa tra tenant su un sistema ONTAP. Dedicare una SVM alle applicazioni consente la delega dei privilegi e l'applicazione di Best practice per limitare il consumo delle risorse.

Sono disponibili diverse opzioni per la gestione di SVM:

- Fornire l'interfaccia di gestione del cluster nella configurazione back-end, insieme alle credenziali appropriate, e specificare il nome SVM.
- Creare un'interfaccia di gestione dedicata per la SVM utilizzando Gestione di sistema di ONTAP o l'interfaccia CLI.
- Condividere il ruolo di gestione con un'interfaccia dati NFS.

In ogni caso, l'interfaccia deve essere in DNS e il nome DNS deve essere utilizzato durante la configurazione di Trident. In questo modo è possibile semplificare alcuni scenari di disaster recovery, ad esempio SVM-DR, senza utilizzare la conservazione delle identità di rete.

Non esiste alcuna preferenza tra avere una LIF di gestione dedicata o condivisa per SVM, tuttavia, è necessario assicurarsi che le policy di sicurezza della rete siano allineate con l'approccio scelto. Indipendentemente da ciò, la LIF di gestione deve essere accessibile tramite DNS per facilitare la massima flessibilità "SVM-DR" Da utilizzare in combinazione con Trident.

Limitare il numero massimo di volumi

I sistemi storage ONTAP hanno un numero massimo di volumi, che varia in base alla versione software e alla piattaforma hardware. Vedere "NetApp Hardware Universe" Per la piattaforma e la versione di ONTAP specifiche per determinare i limiti esatti. Una volta esaurito il numero di volumi, le operazioni di provisioning non vengono eseguite solo per Trident, ma per tutte le richieste di storage.

Di Trident `ontap-nas` e. `ontap-san` I driver forniscono un FlexVolume per ogni volume persistente Kubernetes (PV) creato. Il `ontap-nas-economy` Il driver crea circa un FlexVolume ogni 200 PVS (configurabile tra 50 e 300). Il `ontap-san-economy` Il driver crea circa un FlexVolume ogni 100 PVS (configurabile tra 50 e 200). Per evitare che Trident utilizzi tutti i volumi disponibili sul sistema storage, è necessario impostare un limite per SVM. È possibile eseguire questa operazione dalla riga di comando:

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

Il valore per `max-volumes` varia in base a diversi criteri specifici per l'ambiente:

- Il numero di volumi esistenti nel cluster ONTAP
- Il numero di volumi che si prevede di eseguire il provisioning al di fuori di Trident per altre applicazioni
- Il numero di volumi persistenti che si prevede siano utilizzati dalle applicazioni Kubernetes

Il `max-volumes` Il valore è il totale dei volumi forniti in tutti i nodi del cluster ONTAP e non in un singolo nodo ONTAP. Di conseguenza, potrebbero verificarsi alcune condizioni in cui un nodo del cluster ONTAP potrebbe avere volumi con provisioning Trident molto più o meno elevati rispetto a un altro nodo.

Ad esempio, un cluster ONTAP a due nodi può ospitare un massimo di 2000 FlexVolumes. Il fatto che il numero massimo di volumi sia impostato su 1250 appare molto ragionevole. Tuttavia, se solo "aggregati" Da un nodo vengono assegnati alla SVM oppure gli aggregati assegnati da un nodo non possono essere

sottoposti a provisioning (ad esempio, a causa della capacità), quindi l'altro nodo diventa la destinazione di tutti i volumi con provisioning Trident. Ciò significa che il limite di volume potrebbe essere raggiunto per quel nodo prima di `max-volumes` Viene raggiunto il valore, con conseguente impatto sulle operazioni di Trident e di altri volumi che utilizzano tale nodo. **È possibile evitare questa situazione assicurandosi che gli aggregati di ciascun nodo del cluster siano assegnati alla SVM utilizzata da Trident in numeri uguali.**

Limitare le dimensioni massime dei volumi creati da Trident

Per configurare le dimensioni massime dei volumi che possono essere creati da Trident, utilizzare `limitVolumeSize` nel `backend.json` definizione.

Oltre a controllare le dimensioni del volume nell'array di storage, è necessario sfruttare le funzionalità di Kubernetes.

Configurare Trident per l'utilizzo di CHAP bidirezionale

È possibile specificare i nomi utente e le password dell'iniziatore CHAP e di destinazione nella definizione di backend e impostare Trident per abilitare CHAP su SVM. Utilizzando il `useCHAP` Parametro nella configurazione back-end, Trident autentica le connessioni iSCSI per i backend ONTAP con CHAP. Il supporto CHAP bidirezionale è disponibile con Trident 20.04 e versioni successive.

Creare e utilizzare una policy di QoS SVM

L'utilizzo di una policy di qualità del servizio ONTAP, applicata alla SVM, limita il numero di IOPS consumabili dai volumi sottoposti a provisioning Trident. In questo modo è più utile "prevenire un bullismo" O un container fuori controllo che influisce sui carichi di lavoro al di fuori della SVM Trident.

È possibile creare una policy QoS per SVM in pochi passaggi. Per informazioni più precise, consultare la documentazione relativa alla versione di ONTAP in uso. Nell'esempio riportato di seguito viene creata una policy di QoS che limita a 5000 gli IOPS totali disponibili per la SVM.

```
# create the policy group for the SVM
qos policy-group create -policy-group <policy_name> -vserver <svm_name>
-max-throughput 5000iops

# assign the policy group to the SVM, note this will not work
# if volumes or files in the SVM have existing QoS policies
vserver modify -vserver <svm_name> -qos-policy-group <policy_name>
```

Inoltre, se la tua versione di ONTAP lo supporta, puoi considerare l'utilizzo di un QoS minimo per garantire una quantità di throughput per i carichi di lavoro containerizzati. QoS adattiva non è compatibile con una policy di livello SVM.

Il numero di IOPS dedicati ai carichi di lavoro containerizzati dipende da molti aspetti. Tra le altre cose, queste includono:

- Altri carichi di lavoro che utilizzano lo storage array. Se sono presenti altri carichi di lavoro, non correlati all'implementazione di Kubernetes, che utilizzano le risorse di storage, è necessario prestare attenzione a garantire che tali carichi di lavoro non vengano accidentalmente influenzati negativamente.
- Carichi di lavoro previsti eseguiti in container. Se i carichi di lavoro con requisiti IOPS elevati verranno eseguiti in container, una policy QoS bassa comporta un'esperienza negativa.

È importante ricordare che una policy di QoS assegnata a livello di SVM comporta la condivisione dello stesso pool di IOPS di tutti i volumi forniti a SVM. Se una, o un numero limitato, delle applicazioni containerizzate presenta un elevato requisito di IOPS, potrebbe diventare un problema per gli altri carichi di lavoro containerizzati. In questo caso, è possibile utilizzare l'automazione esterna per assegnare policy QoS per volume.

> È necessario assegnare il gruppo di criteri QoS a SVM **only** se la versione di ONTAP è precedente alla 9.8.

Creare gruppi di policy QoS per Trident

La qualità del servizio (QoS) garantisce che le performance dei carichi di lavoro critici non vengano degradate da carichi di lavoro concorrenti. I gruppi di policy QoS di ONTAP offrono opzioni di QoS per i volumi e consentono agli utenti di definire il limite massimo di throughput per uno o più carichi di lavoro. Per ulteriori informazioni su QoS, vedere "Garanzia di throughput con QoS". È possibile specificare i gruppi di policy QoS nel backend o in un pool di storage, che vengono applicati a ciascun volume creato in quel pool o backend.

ONTAP dispone di due tipi di gruppi di policy QoS: Tradizionale e adattiva. I gruppi di policy tradizionali forniscono un throughput massimo (o minimo, nelle versioni successive) costante negli IOPS. La QoS adattiva scala automaticamente il throughput in base alle dimensioni del carico di lavoro, mantenendo il rapporto tra IOPS e TB|GB in base alle dimensioni del carico di lavoro. Questo offre un vantaggio significativo quando si gestiscono centinaia o migliaia di carichi di lavoro in un'implementazione di grandi dimensioni.

Quando si creano gruppi di criteri QoS, considerare quanto segue:

- Impostare `qosPolicy` digitare `defaults` blocco della configurazione back-end. Vedere il seguente esempio di configurazione del backend:

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "0.0.0.0",
  "dataLIF": "0.0.0.0",
  "svm": "svm0",
  "username": "user",
  "password": "pass",
  "defaults": {
    "qosPolicy": "standard-pg"
  },
  "storage": [
    {
      "labels": {"performance": "extreme"},
      "defaults": {
        "adaptiveQosPolicy": "extremely-adaptive-pg"
      }
    },
    {
      "labels": {"performance": "premium"},
      "defaults": {
        "qosPolicy": "premium-pg"
      }
    }
  ]
}
```

- È necessario applicare i gruppi di criteri per volume, in modo che ogni volume ottenga l'intero throughput come specificato dal gruppo di criteri. I gruppi di criteri condivisi non sono supportati.

Per ulteriori informazioni sui gruppi di criteri QoS, vedere "Comandi QoS di ONTAP 9.8".

Limitare l'accesso alle risorse di storage ai membri del cluster Kubernetes

Limitare l'accesso ai volumi NFS e alle LUN iSCSI create da Trident è un componente critico della posizione di sicurezza per l'implementazione di Kubernetes. In questo modo si impedisce agli host che non fanno parte del cluster Kubernetes di accedere ai volumi e di modificare i dati in modo imprevisto.

È importante comprendere che gli spazi dei nomi sono il limite logico delle risorse in Kubernetes. L'ipotesi è che le risorse nello stesso namespace siano in grado di essere condivise, tuttavia, cosa importante, non esiste alcuna funzionalità di spazio dei nomi incrociato. Ciò significa che anche se i PVS sono oggetti globali, quando sono associati a un PVC sono accessibili solo da pod che si trovano nello stesso namespace. **È fondamentale assicurarsi che gli spazi dei nomi siano utilizzati per fornire la separazione quando appropriato.**

La preoccupazione principale per la maggior parte delle organizzazioni in relazione alla sicurezza dei dati in un contesto Kubernetes è che un processo in un container può accedere allo storage montato sull'host, ma non è destinato al container. "Spazi dei nomi" sono progettati per evitare questo tipo di compromesso. Tuttavia,

esiste un'eccezione: I container con privilegi.

Un container con privilegi è un container che viene eseguito con un numero di autorizzazioni a livello di host sostanzialmente superiore al normale. Per impostazione predefinita, questi elementi non vengono rifiutati, quindi disattivare la funzionalità utilizzando "policy di sicurezza pod".

Per i volumi in cui si desidera accedere sia da Kubernetes che da host esterni, lo storage deve essere gestito in modo tradizionale, con il PV introdotto dall'amministratore e non gestito da Trident. In questo modo, il volume di storage viene distrutto solo quando Kubernetes e gli host esterni si sono disconnessi e non utilizzano più il volume. Inoltre, è possibile applicare una policy di esportazione personalizzata, che consente l'accesso dai nodi del cluster Kubernetes e dai server di destinazione all'esterno del cluster Kubernetes.

Per le implementazioni che hanno nodi di infrastruttura dedicati (ad esempio, OpenShift) o altri nodi che non sono schedulabili per le applicazioni utente, è necessario utilizzare policy di esportazione separate per limitare ulteriormente l'accesso alle risorse di storage. Ciò include la creazione di una policy di esportazione per i servizi implementati nei nodi dell'infrastruttura (ad esempio, i servizi OpenShift Metrics e Logging) e le applicazioni standard implementate nei nodi non dell'infrastruttura.

Utilizzare una policy di esportazione dedicata

È necessario verificare l'esistenza di una policy di esportazione per ciascun backend che consenta l'accesso solo ai nodi presenti nel cluster Kubernetes. Trident può creare e gestire automaticamente le policy di esportazione a partire dalla release 20.04. In questo modo, Trident limita l'accesso ai volumi che fornisce ai nodi nel cluster Kubernetes e semplifica l'aggiunta/eliminazione dei nodi.

In alternativa, è anche possibile creare manualmente una policy di esportazione e compilarla con una o più regole di esportazione che elaborano ogni richiesta di accesso al nodo:

- Utilizzare `vserver export-policy create` Comando ONTAP CLI per creare il criterio di esportazione.
- Aggiungere regole ai criteri di esportazione utilizzando `vserver export-policy rule create` Comando CLI ONTAP.

L'esecuzione di questi comandi consente di limitare i nodi Kubernetes che hanno accesso ai dati.

Disattiva `showmount` Per l'applicazione SVM

Il `showmount` Questa funzione consente a un client NFS di eseguire query su SVM per un elenco delle esportazioni NFS disponibili. Un pod implementato nel cluster Kubernetes può emettere `showmount -e` Eseguire il comando in base al LIF dei dati e ricevere un elenco di montaggi disponibili, inclusi quelli a cui non ha accesso. Sebbene questo, di per sé, non sia un compromesso in termini di sicurezza, fornisce informazioni non necessarie che potrebbero aiutare un utente non autorizzato a connettersi a un'esportazione NFS.

Disattivare `showmount` Utilizzando il comando CLI ONTAP a livello di SVM:

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

Best practice di SolidFire

Scopri le Best practice per la configurazione dello storage SolidFire per Trident.

Crea account SolidFire

Ogni account SolidFire rappresenta un unico proprietario di volume e riceve un proprio set di credenziali CHAP (Challenge-Handshake Authentication Protocol). È possibile accedere ai volumi assegnati a un account utilizzando il nome dell'account e le relative credenziali CHAP o un gruppo di accesso al volume. A un account possono essere assegnati fino a duemila volumi, ma un volume può appartenere a un solo account.

Creare una policy QoS

Utilizzare le policy di qualità del servizio (QoS) di SolidFire se si desidera creare e salvare un'impostazione di qualità del servizio standardizzata che può essere applicata a molti volumi.

È possibile impostare i parametri QoS in base al volume. Le performance per ciascun volume possono essere garantite impostando tre parametri configurabili che definiscono la QoS: Min IOPS, Max IOPS e Burst IOPS.

Di seguito sono riportati i possibili valori IOPS minimi, massimi e burst per la dimensione del blocco di 4 Kb.

Parametro IOPS	Definizione	Min. valore	Valore predefinito	Max. Valore (4 Kb)
IOPS minimi	Il livello garantito di performance per un volume.	50	50	15000
IOPS max	Le performance non supereranno questo limite.	50	15000	200,000
IOPS burst	IOPS massimi consentiti in uno scenario a burst breve.	50	15000	200,000

> Anche se i massimi IOPS e burst IOPS possono essere impostati su 200,000, le performance massime reali di un volume sono limitate dall'utilizzo del cluster e dalle performance per nodo.

Le dimensioni dei blocchi e la larghezza di banda influiscono direttamente sul numero di IOPS. Con l'aumentare delle dimensioni dei blocchi, il sistema aumenta la larghezza di banda fino a raggiungere un livello necessario per elaborare blocchi di dimensioni maggiori. Con l'aumentare della larghezza di banda, il numero di IOPS che il sistema è in grado di raggiungere diminuisce. Vedere "Qualità del servizio SolidFire" Per ulteriori informazioni su QoS e performance.

Autenticazione SolidFire

Element supporta due metodi di autenticazione: CHAP e VAG (Volume Access Group). CHAP utilizza il protocollo CHAP per autenticare l'host nel backend. I gruppi di accesso ai volumi controllano l'accesso ai volumi previsti dall'IT. NetApp consiglia di utilizzare CHAP per l'autenticazione, poiché è più semplice e non ha limiti di scalabilità.

> Trident con il provisioning CSI avanzato supporta l'utilizzo dell'autenticazione CHAP. I VAG devono essere utilizzati solo nella modalità operativa tradizionale non CSI.

L'autenticazione CHAP (verifica che l'iniziatore sia l'utente del volume desiderato) è supportata solo con il

controllo degli accessi basato su account. Se si utilizza CHAP per l'autenticazione, sono disponibili due opzioni: CHAP unidirezionale e CHAP bidirezionale. CHAP unidirezionale autentica l'accesso al volume utilizzando il nome account SolidFire e il segreto dell'iniziatore. L'opzione CHAP bidirezionale rappresenta il metodo più sicuro per autenticare il volume, in quanto il volume autentica l'host tramite il nome account e il segreto dell'iniziatore, quindi l'host autentica il volume tramite il nome account e il segreto di destinazione.

Tuttavia, se non è possibile attivare CHAP e sono richiesti VAG, creare il gruppo di accesso e aggiungere gli iniziatori host e i volumi al gruppo di accesso. Ogni IQN aggiunto a un gruppo di accesso può accedere a ciascun volume del gruppo con o senza autenticazione CHAP. Se iSCSI Initiator è configurato per utilizzare l'autenticazione CHAP, viene utilizzato il controllo degli accessi basato sull'account. Se iSCSI Initiator non è configurato per utilizzare l'autenticazione CHAP, viene utilizzato il controllo di accesso del gruppo di accesso al volume.

Dove trovare ulteriori informazioni?

Di seguito sono elencate alcune delle Best practice. Eseguire una ricerca in "Libreria NetApp" per le versioni più recenti.

ONTAP

- "Guida alle Best practice e all'implementazione di NFS"
- "GUIDA all'amministrazione SAN" (Per iSCSI)
- "Configurazione iSCSI Express per RHEL"

Software Element

- "Configurazione di SolidFire per Linux"

NetApp HCI

- "Prerequisiti per l'implementazione di NetApp HCI"
- "Accedi al NetApp Deployment Engine"

Informazioni sulle Best practice applicative

- "Best practice per MySQL su ONTAP"
- "Best practice per MySQL su SolidFire"
- "NetApp SolidFire e Cassandra"
- "Best practice Oracle su SolidFire"
- "Best practice PostgreSQL su SolidFire"

Non tutte le applicazioni hanno linee guida specifiche, è importante collaborare con il team NetApp e utilizzare "Libreria NetApp" per trovare la documentazione più aggiornata.

Integrare Astra Trident

Per integrare Astra Trident, è necessario integrare i seguenti elementi di progettazione e architettura: Selezione e implementazione dei driver, progettazione della classe di storage, progettazione del pool di storage virtuale, impatto del PVC (Persistent Volume Claim) sul provisioning dello storage, sulle operazioni dei volumi e sull'implementazione

dei servizi OpenShift utilizzando Astra Trident.

Selezione e implementazione dei driver

Scegliere un driver back-end per ONTAP

Per i sistemi ONTAP sono disponibili quattro diversi driver di back-end. Questi driver si differenziano in base al protocollo utilizzato e al modo in cui vengono forniti i volumi nel sistema di storage. Pertanto, prendere in considerazione attentamente il driver da implementare.

A un livello superiore, se l'applicazione dispone di componenti che richiedono storage condiviso (diversi pod che accedono allo stesso PVC), i driver basati su NAS sarebbero la scelta predefinita, mentre i driver iSCSI basati su blocchi soddisfano le esigenze dello storage non condiviso. Scegli il protocollo in base ai requisiti dell'applicazione e al livello di comfort dei team di storage e infrastruttura. In generale, la differenza tra le due applicazioni è minima, quindi spesso la decisione si basa sulla necessità o meno di uno storage condiviso (in cui più di un pod necessitano di accesso simultaneo).

Di seguito sono elencati i cinque driver per i backend ONTAP:

- `ontap-nas`: Ogni PV fornito è un FlexVolume ONTAP completo.
- `ontap-nas-economy`: Ogni PV fornito è un qtree, con un numero configurabile di qtree per FlexVolume (il valore predefinito è 200).
- `ontap-nas-flexgroup`: Vengono utilizzati tutti i PV forniti come ONTAP FlexGroup completo e tutti gli aggregati assegnati a una SVM.
- `ontap-san`: Ogni PV fornito è un LUN all'interno del proprio FlexVolume.
- `ontap-san-economy`: Ogni PV fornito è un LUN, con un numero configurabile di LUN per FlexVolume (il valore predefinito è 100).

La scelta tra i tre driver NAS ha alcune ramificazioni alle funzionalità, che sono rese disponibili per l'applicazione.

Si noti che, nelle tabelle seguenti, non tutte le funzionalità sono esposte attraverso Astra Trident. Alcuni devono essere applicati dall'amministratore dello storage dopo il provisioning, se si desidera questa funzionalità. Le note a piè di pagina in superscript distinguono le funzionalità per funzionalità e driver.

Driver NAS ONTAP	Snapshot	Cloni	Policy di esportazione dinamiche	Multi-attach	QoS	Ridimensionare	Replica
`ontap-nas`	Sì	Sì	Yes [5]	Sì	Yes [1]	Sì	Yes [1]
`ontap-nas-economy`	Yes [3]	Yes [3]	Yes [5]	Sì	Yes [3]	Sì	Yes [3]
`ontap-nas-flexgroup`	Yes [1]	No	Yes [5]	Sì	Yes [1]	Sì	Yes [1]

Astra Trident offre 2 driver SAN per ONTAP, le cui funzionalità sono illustrate di seguito.

Driver SAN ONTAP	Snapshot	Cloni	Multi-attach	CHAP bidirezionale	QoS	Ridimensionare	Replica
`ontap-san`	Sì	Sì	Yes [4]	Sì	Yes [1]	Sì	Yes [1]
`ontap-san-economy`	Sì	Sì	Yes [4]	Sì	Yes [3]	Yes [1]	Yes [3]

> Nota a piè di pagina per le tabelle precedenti: Yes [1]: Non gestito da Astra Trident Yes [2]: Gestito da Astra Trident, ma non da PV Granular Yesnota a piè di pagina:3[]: Non gestito da Astra Trident e non da PV Granular Yesnota a piè di pagina:4[]: Supportato da CSI Trident

Le funzionalità non granulari PV vengono applicate all'intero FlexVolume e tutti i PVS (ovvero qtree o LUN in FlexVol condivisi) condividono una pianificazione comune.

Come si può vedere nelle tabelle precedenti, gran parte delle funzionalità tra `ontap-nas` e. `ontap-nas-economy` è lo stesso. Tuttavia, perché il `ontap-nas-economy` Driver limita la capacità di controllare la pianificazione in base alla granularità per PV, questo può influire in particolare sul disaster recovery e sulla pianificazione del backup. Per i team di sviluppo che desiderano sfruttare la funzionalità dei cloni PVC sullo storage ONTAP, ciò è possibile solo quando si utilizza `ontap-nas`, `ontap-san` oppure `ontap-san-economy` driver.

> Il `solidfire-san` Il driver è anche in grado di clonare i PVC.

Scegliere un driver back-end per Cloud Volumes ONTAP

Cloud Volumes ONTAP offre il controllo dei dati e funzionalità di storage di livello Enterprise per diversi casi di utilizzo, tra cui condivisioni di file e storage a livello di blocco che servono protocolli NAS e SAN (NFS, SMB/CIFS e iSCSI). I driver compatibili per Cloud Volume ONTAP sono `ontap-nas`, `ontap-nas-economy`, `ontap-san` e. `ontap-san-economy`. Questi sono validi per Cloud Volume ONTAP per Azure, Cloud Volume ONTAP per GCP.

Scegli un driver back-end per Amazon FSX per ONTAP

Amazon FSX per ONTAP consente ai clienti di sfruttare le funzionalità, le performance e le funzionalità amministrative di NetApp che conoscono, sfruttando al contempo la semplicità, l'agilità, la sicurezza e la scalabilità dell'archiviazione dei dati su AWS. FSX per ONTAP supporta molte delle funzionalità del file system e delle API di amministrazione di ONTAP. I driver compatibili per Cloud Volume ONTAP sono `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san` e. `ontap-san-economy`.

Scegliere un driver back-end per NetApp HCI/SolidFire

Il `solidfire-san` Il driver utilizzato con le piattaforme NetApp HCI/SolidFire aiuta l'amministratore a configurare un backend elemento per Trident in base ai limiti di QoS. Se si desidera progettare il backend per impostare i limiti di QoS specifici sui volumi forniti da Trident, utilizzare `type` nel file backend. L'amministratore può inoltre limitare le dimensioni del volume che è possibile creare sullo storage utilizzando `limitVolumeSize` parametro. Attualmente, le funzionalità di storage degli elementi come il ridimensionamento del volume e la replica del volume non sono supportate da `solidfire-san` driver. Queste operazioni devono essere eseguite manualmente tramite l'interfaccia utente Web di Element Software.

Driver SolidFire	Snapshot	Cloni	Multi-attach	CAP	QoS	Ridimensionare	Replica
`solidfire-san`	Sì	Sì	Yes [2]	Sì	Sì	Sì	Yes [1]

> Nota a piè di pagina: Yesnota a piè di pagina:1[]: Non gestito da Astra Trident
> Yesnota a piè di pagina:2[]: Supportato per i volumi raw-block

Scegliere un driver back-end per Azure NetApp Files

Astra Trident utilizza `azure-netapp-files` driver per la gestione di "Azure NetApp Files" servizio.

Per ulteriori informazioni su questo driver e su come configurarlo, consultare "Configurazione backend Astra Trident per Azure NetApp Files".

Driver Azure NetApp Files	Snapshot	Cloni	Multi-attach	QoS	Espandere	Replica
`azure-netapp-files`	Sì	Sì	Sì	Sì	Sì	Yes [1]

> Nota a piè di pagina: Yesnota a piè di pagina:1[]: Non gestita da Astra Trident

Scegli un driver back-end per Cloud Volumes Service con GCP

Astra Trident utilizza `gcp-cvs` Driver per il collegamento con Cloud Volumes Service sul backend GCP. Per configurare il backend GCP su Trident, è necessario specificare `projectNumber`, `apiRegion`, e. `apiKey` nel file backend. Il numero del progetto si trova nel portale Web GCP, mentre la chiave API deve essere presa dal file della chiave privata dell'account del servizio creato durante la configurazione dell'accesso API per i volumi cloud su GCP. Astra Trident può creare volumi CVS in uno dei due "tipi di servizio":

1. **CVS**: Il tipo di servizio CVS di base, che offre un'elevata disponibilità zonale con livelli di performance limitati/moderati.
2. **CVS-Performance**: Tipo di servizio ottimizzato per le performance più adatto per i carichi di lavoro di produzione che apprezzano le performance. Scegli tra tre livelli di servizio unici [`standard`, `premium`, e. `extreme`]. Attualmente, 100 GiB è la dimensione minima del volume CVS-Performance che verrà fornito, mentre i volumi CVS devono essere almeno 300 GiB. Le versioni future di CVS potrebbero rimuovere questa restrizione.

> Quando si implementano backend utilizzando il tipo di servizio CVS predefinito [`storageClass=software`], gli utenti **devono ottenere l'accesso** alla funzione volumi sub-1TiB su GCP per i numeri di progetto e gli ID progetto in questione. Ciò è necessario per Trident per eseguire il provisioning di volumi inferiori a 1 TiB. In caso contrario, le creazioni dei volumi **non avranno esito positivo** per i PVC con meno di 600 GiB. Utilizzare "questo modulo" Per ottenere l'accesso a volumi inferiori a 1 TiB.

CVS per driver GCP	Snapshot	Cloni	Multi-attach	QoS	Espandere	Replica
`gcp-cvs`	Sì	Sì	Sì	Sì	Sì	Yes [1]

> Nota a piè di pagina: Yesnota a piè di pagina:1[]: Non gestita da Astra Trident

Il `gcp-cvs` il driver utilizza pool di storage virtuali. I pool di storage virtuali astraggono il backend, consentendo ad Astra Trident di decidere il posizionamento dei volumi. L'amministratore definisce i pool di storage virtuali nei file backend.json. Le classi di storage identificano i pool di storage virtuali utilizzando le etichette.

Design di classe storage

È necessario configurare e applicare singole classi di storage per creare un oggetto Kubernetes Storage Class. In questa sezione viene descritto come progettare una classe di storage per l'applicazione.

Design di classe storage per un utilizzo specifico del back-end

Il filtraggio può essere utilizzato all'interno di un oggetto specifico della classe di storage per determinare quale pool o insieme di pool di storage utilizzare con tale classe di storage specifica. Nella classe di storage è possibile impostare tre set di filtri: `storagePools`, `additionalStoragePools`, e/o. `excludeStoragePools`.

Il `storagePools` parametro consente di limitare lo storage al set di pool che corrispondono a qualsiasi attributo specificato. Il `additionalStoragePools` Il parametro viene utilizzato per estendere il set di pool che Astra Trident utilizzerà per il provisioning insieme al set di pool selezionato dagli attributi e. `storagePools` parametri. È possibile utilizzare i parametri singolarmente o entrambi insieme per assicurarsi che sia selezionato il set appropriato di pool di storage.

Il `excludeStoragePools` il parametro viene utilizzato per escludere in modo specifico il set di pool elencato che corrispondono agli attributi.

Design di classe storage per emulare le policy QoS

Se si desidera progettare classi di storage per emulare le policy di qualità del servizio, creare una classe di storage con `media` attributo come `hdd` oppure `ssd`. Basato su `media` Attributo menzionato nella classe di storage, Trident selezionerà il backend appropriato che serve `hdd` oppure `ssd` aggregato in modo da corrispondere all'attributo di supporto e indirizzare il provisioning dei volumi sull'aggregato specifico. Pertanto, possiamo creare una classe di storage PREMIUM che avrebbe `media` attributo impostato come `ssd` Che potrebbero essere classificati come policy DI qualità del servizio PREMIUM. È possibile creare un altro STANDARD di classe storage con l'attributo media impostato come `hdd' che potrebbe essere classificato come policy standard di QoS. Potremmo anche utilizzare l'attributo ``IOPS" nella classe di storage per reindirizzare il provisioning a un'appliance Element che può essere definita come policy QoS.

Design di classe storage per utilizzare il back-end in base a funzionalità specifiche

Le classi di storage possono essere progettate per indirizzare il provisioning dei volumi su un backend specifico in cui sono abilitate funzionalità come thin provisioning e thick provisioning, snapshot, cloni e crittografia. Per specificare lo storage da utilizzare, creare classi di storage che specifichino il backend appropriato con la funzionalità richiesta attivata.

Design di classe storage per i pool di storage virtuali

I pool di storage virtuali sono disponibili per tutti i backend Astra Trident. È possibile definire Virtual Storage Pools per qualsiasi backend, utilizzando qualsiasi driver fornito da Astra Trident.

I pool di storage virtuali consentono a un amministratore di creare un livello di astrazione sui backend a cui si può fare riferimento attraverso le classi di storage, per una maggiore flessibilità e un posizionamento efficiente dei volumi sui backend. È possibile definire backend diversi con la stessa classe di servizio. Inoltre, è possibile creare più pool di storage sullo stesso backend, ma con caratteristiche diverse. Quando una classe di storage viene configurata con un selettore con le etichette specifiche, Astra Trident sceglie un backend che

corrisponde a tutte le etichette del selettore per posizionare il volume. Se le etichette del selettore Storage Class corrispondono a più Storage Pools, Astra Trident sceglierà una di queste da cui eseguire il provisioning del volume.

Progettazione del pool di storage virtuale

Durante la creazione di un backend, in genere è possibile specificare un set di parametri. Per l'amministratore non era possibile creare un altro backend con le stesse credenziali di storage e con un set di parametri diverso. Con l'introduzione dei Virtual Storage Pools, questo problema è stato risolto. Virtual Storage Pools è un'astrazione di livello introdotta tra il backend e Kubernetes Storage Class, in modo che l'amministratore possa definire i parametri insieme alle etichette a cui si può fare riferimento attraverso le classi di storage di Kubernetes come un selettore, in modo indipendente dal backend. È possibile definire i pool di storage virtuali per tutti i backend NetApp supportati con Astra Trident. L'elenco include SolidFire/NetApp HCI, ONTAP, Cloud Volumes Service su GCP e Azure NetApp Files.

> Quando si definiscono i pool di storage virtuali, si consiglia di non tentare di riorganizzare l'ordine dei pool virtuali esistenti in una definizione di backend. Si consiglia inoltre di non modificare/modificare gli attributi di un pool virtuale esistente e di non definire un nuovo pool virtuale.

Progettare Virtual Storage Pools per emulare diversi livelli di servizio/QoS

È possibile progettare Virtual Storage Pools per emulare le classi di servizio. Utilizzando l'implementazione del pool virtuale per il servizio volume cloud per Azure NetApp Files, esaminiamo come possiamo configurare diverse classi di servizio. Configurare il backend ANF con più etichette, che rappresentano diversi livelli di performance. Impostare `servicelevel` aspect al livello di performance appropriato e aggiungere altri aspetti richiesti sotto ogni etichetta. Creare ora diverse classi di storage Kubernetes che si mappano a diversi pool di storage virtuali. Utilizzando il `parameters.selector` Ciascun StorageClass richiama i pool virtuali che possono essere utilizzati per ospitare un volume.

Progettare i Virtual Pools per assegnare un insieme specifico di aspetti

È possibile progettare più pool di storage virtuali con un insieme specifico di aspetti da un singolo backend di storage. A tale scopo, configurare il backend con più etichette e impostare gli aspetti richiesti sotto ciascuna etichetta. Ora è possibile creare diverse classi di storage Kubernetes utilizzando `parameters.selector` Campo che viene mappato a diversi pool di storage virtuali. I volumi con provisioning sul back-end avranno gli aspetti definiti nel Virtual Storage Pool scelto.

Caratteristiche del PVC che influiscono sul provisioning dello storage

Alcuni parametri oltre la classe di storage richiesta possono influire sul processo decisionale di provisioning di Astra Trident durante la creazione di un PVC.

Modalità di accesso

Quando si richiede lo storage tramite PVC, uno dei campi obbligatori è la modalità di accesso. La modalità desiderata può influire sul backend selezionato per ospitare la richiesta di storage.

Astra Trident tenterà di associare il protocollo di storage utilizzato al metodo di accesso specificato in base alla matrice seguente. Ciò è indipendente dalla piattaforma di storage sottostante.

	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
ISCSI	Sì	Sì	Sì (blocco raw)
NFS	Sì	Sì	Sì

Una richiesta di ReadWriteMany PVC inviata a un'implementazione Trident senza un backend NFS configurato non comporterà il provisioning di alcun volume. Per questo motivo, il richiedente deve utilizzare la modalità di accesso appropriata per la propria applicazione.

Operazioni di volume

Modificare i volumi persistenti

I volumi persistenti sono, con due eccezioni, oggetti immutabili in Kubernetes. Una volta creata, la policy di recupero e le dimensioni possono essere modificate. Tuttavia, questo non impedisce che alcuni aspetti del volume vengano modificati al di fuori di Kubernetes. Ciò può essere utile per personalizzare il volume per applicazioni specifiche, per garantire che la capacità non venga accidentalmente consumata o semplicemente per spostare il volume in un controller di storage diverso per qualsiasi motivo.

> Attualmente, i provisioning in-tree di Kubernetes non supportano le operazioni di ridimensionamento dei volumi per NFS o iSCSI PVS. Astra Trident supporta l'espansione dei volumi NFS e iSCSI.

I dettagli di connessione del PV non possono essere modificati dopo la creazione.

Creazione di snapshot di volumi on-demand

Astra Trident supporta la creazione on-demand di snapshot di volumi e la creazione di PVC da snapshot utilizzando il framework CSI. Gli snapshot offrono un metodo pratico per mantenere copie point-in-time dei dati e hanno un ciclo di vita indipendente dal PV di origine in Kubernetes. Queste snapshot possono essere utilizzate per clonare i PVC.

Creare volumi da snapshot

Astra Trident supporta anche la creazione di PersistentVolumes da snapshot di volumi. A tale scopo, è sufficiente creare un PersistentVolumeClaim e citare il `datasource` come snapshot richiesto da cui è necessario creare il volume. Astra Trident gestirà questo PVC creando un volume con i dati presenti nello snapshot. Con questa funzionalità, è possibile duplicare i dati tra regioni, creare ambienti di test, sostituire un volume di produzione danneggiato o corrotto nella sua interezza o recuperare file e directory specifici e trasferirli in un altro volume collegato.

Spostare i volumi nel cluster

Gli amministratori dello storage hanno la possibilità di spostare i volumi tra aggregati e controller nel cluster ONTAP senza interruzioni per il consumatore di storage. Questa operazione non influisce su Astra Trident o sul cluster Kubernetes, purché l'aggregato di destinazione sia un aggregato a cui ha accesso la SVM utilizzata da Astra Trident. Cosa importante, se l'aggregato è stato aggiunto di recente alla SVM, il backend dovrà essere aggiornato aggiungendolo nuovamente ad Astra Trident. In questo modo Astra Trident reinventarierà la SVM in modo che il nuovo aggregato venga riconosciuto.

Tuttavia, Astra Trident non supporta automaticamente lo spostamento dei volumi tra backend. Ciò include le SVM nello stesso cluster, tra cluster o su una piattaforma storage diversa (anche se il sistema storage è collegato ad Astra Trident).

Se un volume viene copiato in un'altra posizione, la funzione di importazione del volume può essere utilizzata per importare i volumi correnti in Astra Trident.

Espandere i volumi

Astra Trident supporta il ridimensionamento di NFS e iSCSI PVS. Ciò consente agli utenti di ridimensionare i propri volumi direttamente attraverso il livello Kubernetes. L'espansione dei volumi è possibile per tutte le principali piattaforme di storage NetApp, inclusi i backend ONTAP, SolidFire/NetApp HCI e Cloud Volumes Service. Per consentire la possibile espansione in un secondo momento, impostare `allowVolumeExpansion` a. `true` Nel StorageClass associato al volume. Ogni volta che è necessario ridimensionare il volume persistente, modificare `spec.resources.requests.storage` Annotazione nella richiesta di rimborso del volume persistente sulla dimensione del volume richiesta. Trident si occuperà utomaticamente del ridimensionamento del volume sul cluster di storage.

Importare un volume esistente in Kubernetes

L'importazione dei volumi consente di importare un volume di storage esistente in un ambiente Kubernetes. Questa funzione è attualmente supportata da `ontap-nas`, `ontap-nas-flexgroup`, `solidfire-san`, `azure-netapp-files`, e. `gcp-cvs` driver. Questa funzionalità è utile quando si esegue il porting di un'applicazione esistente in Kubernetes o durante scenari di disaster recovery.

Quando si utilizza ONTAP e. `solidfire-san` driver, utilizzare il comando `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` Per importare un volume esistente in Kubernetes da gestire da Astra Trident. Il file PVC YAML o JSON utilizzato nel comando del volume di importazione punta a una classe di storage che identifica Astra Trident come provider. Quando si utilizza un backend NetApp HCI/SolidFire, assicurarsi che i nomi dei volumi siano univoci. Se i nomi dei volumi sono duplicati, clonare il volume con un nome univoco in modo che la funzione di importazione dei volumi possa distinguerli.

Se il `azure-netapp-files` oppure `gcp-cvs` driver, utilizzare il comando `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` Importare il volume in Kubernetes da gestire da Astra Trident. In questo modo si garantisce un riferimento di volume univoco.

Quando viene eseguito il comando precedente, Astra Trident troverà il volume sul backend e ne leggerà le dimensioni. Aggiunge automaticamente (e sovrascrive se necessario) le dimensioni del volume PVC configurato. Astra Trident crea quindi il nuovo PV e Kubernetes lega il PVC al PV.

Se un container fosse stato implementato in modo da richiedere lo specifico PVC importato, rimarrebbe in sospeso fino a quando la coppia PVC/PV non sarà legata tramite il processo di importazione del volume. Una volta rilegata la coppia PVC/PV, il container dovrebbe salire, a condizione che non vi siano altri problemi.

Implementare i servizi OpenShift

I servizi cluster OpenShift a valore aggiunto offrono funzionalità importanti agli amministratori dei cluster e alle applicazioni ospitate. Lo storage utilizzato da questi servizi può essere fornito utilizzando le risorse locali del nodo, tuttavia, questo spesso limita la capacità, le performance, la ripristinabilità e la sostenibilità del servizio. Sfruttando un array di storage Enterprise per fornire la capacità a questi servizi è possibile migliorare drasticamente il servizio, tuttavia, come per tutte le applicazioni, OpenShift e gli amministratori dello storage dovrebbero collaborare strettamente per determinare le opzioni migliori per ciascuno di essi. La documentazione di Red Hat deve essere sfruttata in maniera significativa per determinare i requisiti e garantire che le esigenze di dimensionamento e performance siano soddisfatte.

Servizio di registro

La distribuzione e la gestione dello storage per il registro sono state documentate su "netapp.io" in "blog".

Servizio di registrazione

Come gli altri servizi OpenShift, il servizio di logging viene implementato utilizzando Ansible con parametri di configurazione forniti dal file di inventario, ovvero host, forniti al playbook. Sono previsti due metodi di installazione: Distribuzione del logging durante l'installazione iniziale di OpenShift e distribuzione del logging dopo l'installazione di OpenShift.

> A partire dalla versione 3.9 di Red Hat OpenShift, la documentazione ufficiale consiglia NFS per il servizio di logging a causa di problemi legati alla corruzione dei dati. Questo si basa sui test Red Hat dei loro prodotti. Il server NFS di ONTAP non presenta questi problemi e può facilmente eseguire il backup di un'implementazione di logging. In definitiva, la scelta del protocollo per il servizio di logging dipende da voi, sappiate che entrambi funzioneranno benissimo quando si utilizzano le piattaforme NetApp e che non vi è alcun motivo per evitare NFS se questa è la vostra preferenza.

Se si sceglie di utilizzare NFS con il servizio di registrazione, è necessario impostare la variabile Ansible `openshift_enable_unsupported_configurations` a. `true` per impedire il malfunzionamento del programma di installazione.

Inizia subito

Il servizio di logging può, facoltativamente, essere implementato per entrambe le applicazioni e per le operazioni principali del cluster OpenShift stesso. Se si sceglie di implementare la registrazione delle operazioni, specificando la variabile `openshift_logging_use_ops` come `true`, verranno create due istanze del servizio. Le variabili che controllano l'istanza di logging per le operazioni contengono "Ops" al loro interno, mentre l'istanza per le applicazioni non lo fa.

La configurazione delle variabili Ansible in base al metodo di implementazione è importante per garantire che lo storage corretto venga utilizzato dai servizi sottostanti. Diamo un'occhiata alle opzioni per ciascuno dei metodi di implementazione.

> Le tabelle seguenti contengono solo le variabili rilevanti per la configurazione dello storage in relazione al servizio di registrazione. Altre opzioni sono disponibili in "Documentazione di registrazione di RedHat OpenShift" che devono essere esaminate, configurate e utilizzate in base all'implementazione.

Le variabili riportate nella tabella seguente determineranno la creazione di un PV e di un PVC per il servizio di registrazione utilizzando i dettagli forniti. Questo metodo è notevolmente meno flessibile rispetto all'utilizzo del playbook di installazione dei componenti dopo l'installazione di OpenShift, tuttavia, se si dispone di volumi esistenti, si tratta di un'opzione.

Variabile	Dettagli
`openshift_logging_storage_kind`	Impostare su `nfs` Per fare in modo che il programma di installazione crei un NFS PV per il servizio di registrazione.
`openshift_logging_storage_host`	Il nome host o l'indirizzo IP dell'host NFS. Questa opzione deve essere impostata sul LIF dei dati per la macchina virtuale.

Variabile	Dettagli
`openshift_logging_storage_nfs_directory`	Il percorso di montaggio per l'esportazione NFS. Ad esempio, se il volume è giuntato come `/openshift_logging`, utilizzare tale percorso per questa variabile.
`openshift_logging_storage_volume_name`	Il nome, ad esempio `pv_ose_logs`, Del PV da creare.
`openshift_logging_storage_volume_size`	Le dimensioni dell'esportazione NFS, ad esempio `100Gi`.

Se il cluster OpenShift è già in esecuzione e quindi Trident è stato implementato e configurato, l'installatore può utilizzare il provisioning dinamico per creare i volumi. È necessario configurare le seguenti variabili.

Variabile	Dettagli
`openshift_logging_es_pvc_dynamic`	Impostare su true per utilizzare volumi con provisioning dinamico.
`openshift_logging_es_pvc_storage_class_name`	Il nome della classe di storage che verrà utilizzata nel PVC.
`openshift_logging_es_pvc_size`	La dimensione del volume richiesto nel PVC.
`openshift_logging_es_pvc_prefix`	Prefisso dei PVC utilizzati dal servizio di registrazione.
`openshift_logging_es_ops_pvc_dynamic`	Impostare su `true` per utilizzare volumi con provisioning dinamico per l'istanza di logging ops.
`openshift_logging_es_ops_pvc_storage_class_name`	Il nome della classe di storage per l'istanza di logging di Ops.
`openshift_logging_es_ops_pvc_size`	La dimensione della richiesta di volume per l'istanza Ops.
`openshift_logging_es_ops_pvc_prefix`	Un prefisso per i PVC di istanza di Ops.

Implementare lo stack di logging

Se si sta implementando la registrazione come parte del processo di installazione iniziale di OpenShift, è sufficiente seguire il processo di distribuzione standard. Ansible configurerà e implementerà i servizi e gli oggetti OpenShift necessari in modo che il servizio sia disponibile non appena Ansible sarà completato.

Tuttavia, se si esegue l'implementazione dopo l'installazione iniziale, Ansible dovrà utilizzare il playbook dei componenti. Questo processo potrebbe cambiare leggermente con diverse versioni di OpenShift, quindi assicurati di leggere e seguire "Documentazione di RedHat OpenShift Container Platform 3.11" per la versione in uso.

Servizio di metriche

Il servizio Metrics fornisce all'amministratore informazioni preziose sullo stato, l'utilizzo delle risorse e la disponibilità del cluster OpenShift. È inoltre necessario per la funzionalità di scalabilità automatica del pod e molte organizzazioni utilizzano i dati del servizio di metriche per le proprie applicazioni di riaccredito e/o visualizzazione.

Come nel caso del servizio di registrazione e di OpenShift nel suo complesso, Ansible viene utilizzato per implementare il servizio di metriche. Inoltre, come il servizio di registrazione, il servizio di metriche può essere implementato durante una configurazione iniziale del cluster o dopo che è operativo utilizzando il metodo di installazione dei componenti. Le seguenti tabelle contengono le variabili importanti per la configurazione dello storage persistente per il servizio di metriche.

> Le tabelle seguenti contengono solo le variabili rilevanti per la configurazione dello storage in relazione al servizio di metriche. La documentazione contiene molte altre opzioni che devono essere esaminate, configurate e utilizzate in base all'implementazione.

Variabile	Dettagli
`openshift_metrics_storage_kind`	Impostare su `nfs` Per fare in modo che il programma di installazione crei un NFS PV per il servizio di registrazione.
`openshift_metrics_storage_host`	Il nome host o l'indirizzo IP dell'host NFS. Questa opzione deve essere impostata sul valore LIF dei dati per SVM.
`openshift_metrics_storage_nfs_directory`	Il percorso di montaggio per l'esportazione NFS. Ad esempio, se il volume è giuntato come `/openshift_metrics`, utilizzare tale percorso per questa variabile.
`openshift_metrics_storage_volume_name`	Il nome, ad esempio `pv_ose_metrics`, Del PV da creare.
`openshift_metrics_storage_volume_size`	Le dimensioni dell'esportazione NFS, ad esempio `100Gi`.

Se il cluster OpenShift è già in esecuzione e quindi Trident è stato implementato e configurato, l'installatore può utilizzare il provisioning dinamico per creare i volumi. È necessario configurare le seguenti variabili.

Variabile	Dettagli
`openshift_metrics_cassandra_pvc_prefix`	Prefisso da utilizzare per i PVC di metriche.
`openshift_metrics_cassandra_pvc_size`	Le dimensioni dei volumi da richiedere.
`openshift_metrics_cassandra_storage_type`	Il tipo di storage da utilizzare per le metriche, deve essere impostato su dinamico per Ansible per creare PVC con la classe di storage appropriata.
`openshift_metrics_cassanda_pvc_storage_class_name`	Il nome della classe di storage da utilizzare.

Implementare il servizio di metriche

Con le variabili Ansible appropriate definite nel file di host/inventario, implementare il servizio utilizzando Ansible. Se si esegue l'implementazione al momento dell'installazione di OpenShift, il PV verrà creato e utilizzato automaticamente. Se si esegue l'implementazione utilizzando i playbook dei componenti, dopo l'installazione di OpenShift, Ansible creerà tutti i PVC necessari e, dopo che Astra Trident ha eseguito il provisioning dello storage, implementerà il servizio.

Le variabili di cui sopra e il processo di implementazione possono cambiare con ogni versione di OpenShift.

Verifica e segui "Guida all'implementazione di OpenShift di RedHat" per la versione in uso, in modo che sia configurata per l'ambiente in uso.

Protezione dei dati

Scopri le opzioni di ripristino e protezione dei dati fornite dalle piattaforme storage di NetApp. Astra Trident può eseguire il provisioning di volumi che possono sfruttare alcune di queste funzionalità. È necessario disporre di una strategia di protezione e ripristino dei dati per ogni applicazione con un requisito di persistenza.

Eseguire il backup di `etcd` dati del cluster

Astra Trident memorizza i propri metadati nel cluster Kubernetes `etcd` database. Eseguire periodicamente il backup di `etcd` I dati del cluster sono importanti per ripristinare i cluster Kubernetes in situazioni di emergenza.

Fasi

1. Il `etcdctl snapshot save` il comando consente di acquisire un'istantanea point-in-time di `etcd` cluster:

   ```
   sudo docker run --rm -v /backup:/backup \
     --network host \
     -v /etc/kubernetes/pki/etcd:/etc/kubernetes/pki/etcd \
     --env ETCDCTL_API=3 \
     k8s.gcr.io/etcd-amd64:3.2.18 \
     etcdctl --endpoints=https://127.0.0.1:2379 \
     --cacert=/etc/kubernetes/pki/etcd/ca.crt \
     --cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt \
     --key=/etc/kubernetes/pki/etcd/healthcheck-client.key \
     snapshot save /backup/etcd-snapshot.db
   ```

   Questo comando crea uno snapshot etcd creando un container etcd e salvandolo in `/backup` directory.

2. In caso di disastro, è possibile accelerare un cluster Kubernetes utilizzando le snapshot etcd. Utilizzare `etcdctl snapshot restore` comando per ripristinare uno snapshot specifico su `/var/lib/etcd` cartella. Dopo il ripristino, verificare se `/var/lib/etcd` la cartella è stata popolata con `member` cartella. Di seguito viene riportato un esempio di `etcdctl snapshot restore` comando:

   ```
   # etcdctl snapshot restore '/backup/etcd-snapshot-latest.db' ; mv
   /default.etcd/member/ /var/lib/etcd/
   ```

3. Prima di inizializzare il cluster Kubernetes, copiare tutti i certificati necessari.
4. Creare il cluster con `--ignore-preflight-errors=DirAvailable—var-lib-etcd` allarme.
5. Una volta attivato il cluster, assicurarsi che i pod del sistema kube siano stati avviati.
6. Utilizzare `kubectl get crd` Per verificare se le risorse personalizzate create da Trident sono presenti e recuperare gli oggetti Trident per assicurarsi che tutti i dati siano disponibili.

Ripristinare la data utilizzando le snapshot ONTAP

Le snapshot svolgono un ruolo importante fornendo opzioni di recovery point-in-time per i dati delle applicazioni. Tuttavia, gli snapshot non sono backup da soli, ma non proteggono da guasti del sistema di storage o altre catastrofi. Tuttavia, rappresentano un metodo pratico, rapido e semplice per ripristinare i dati nella maggior parte degli scenari. Scopri come utilizzare la tecnologia ONTAP Snapshot per eseguire backup del volume e come ripristinarli.

- Se il criterio di snapshot non è stato definito nel backend, per impostazione predefinita viene utilizzato il `none` policy. In questo modo, ONTAP non crea snapshot automatiche. Tuttavia, l'amministratore dello storage può eseguire snapshot manuali o modificare il criterio di snapshot tramite l'interfaccia di gestione di ONTAP. Ciò non influisce sul funzionamento di Trident.
- La directory di snapshot è nascosta per impostazione predefinita. In questo modo, è possibile semplificare la massima compatibilità dei volumi con provisioning tramite `ontap-nas` e. `ontap-nas-economy` driver. Attivare il `.snapshot` directory quando si utilizza `ontap-nas` e. `ontap-nas-economy` driver per consentire alle applicazioni di ripristinare direttamente i dati dalle snapshot.
- Ripristinare uno stato di un volume registrato in uno snapshot precedente utilizzando `volume snapshot restore` Comando CLI ONTAP. Quando si ripristina una copia snapshot, l'operazione di ripristino sovrascrive la configurazione del volume esistente. Tutte le modifiche apportate ai dati nel volume dopo la creazione della copia Snapshot andranno perse.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

Replicare i dati utilizzando ONTAP

La replica dei dati può svolgere un ruolo importante nella protezione contro la perdita di dati dovuta a guasti degli array di storage.

> Per ulteriori informazioni sulle tecnologie di replica di ONTAP, vedere "Documentazione ONTAP".

Replica di SnapMirror Storage Virtual Machine (SVM)

È possibile utilizzare "SnapMirror" Replicare una SVM completa, che include le impostazioni di configurazione e i volumi. In caso di disastro, è possibile attivare la SVM di destinazione di SnapMirror per iniziare a fornire i dati. Una volta ripristinati i sistemi, è possibile tornare al sistema primario.

Astra Trident non è in grado di configurare le relazioni di replica, pertanto l'amministratore dello storage può utilizzare la funzione di replica SVM di SnapMirror di ONTAP per replicare automaticamente i volumi in una destinazione di disaster recovery (DR).

Considerare quanto segue se si intende utilizzare la funzione di replica SVM di SnapMirror o se si sta utilizzando la funzione:

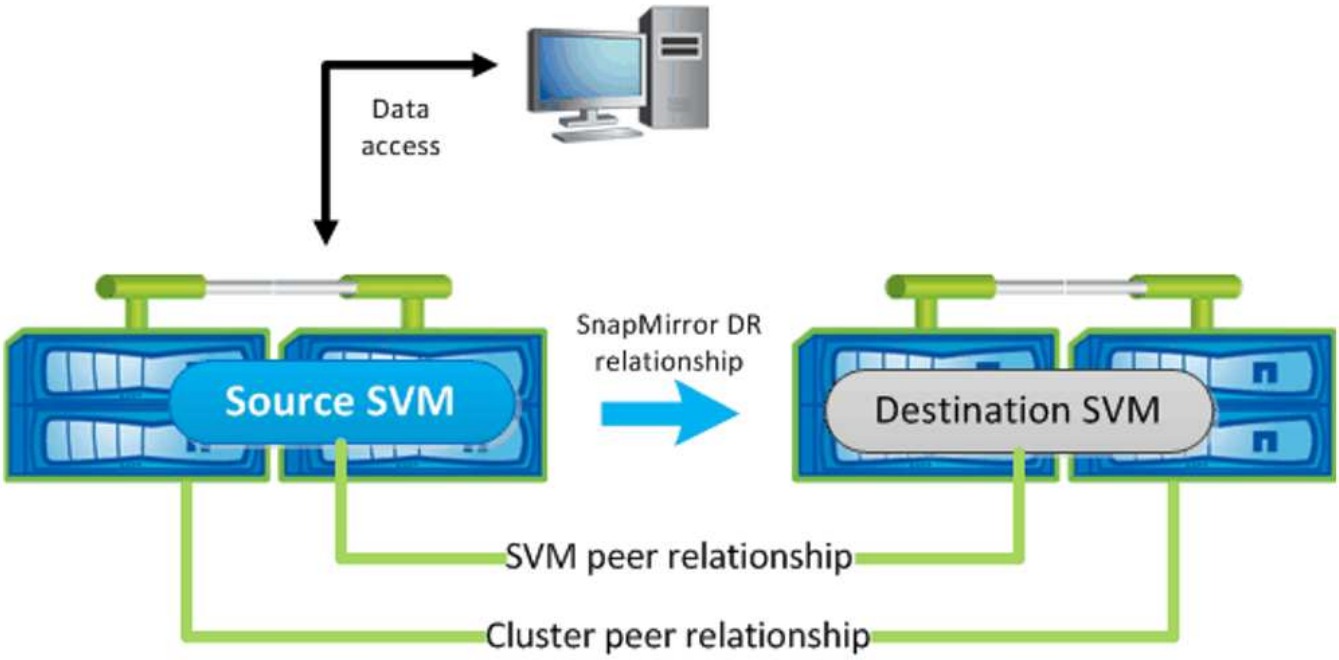
- È necessario creare un backend distinto per ogni SVM, che ha SVM-DR abilitato.
- È necessario configurare le classi di storage in modo da non selezionare i backend replicati, tranne quando si desidera. Ciò è importante per evitare di disporre di volumi che non richiedono la protezione di una relazione di replica per il provisioning sul back-end che supporta SVM-DR.
- Gli amministratori delle applicazioni devono comprendere i costi e la complessità aggiuntivi associati alla

replica dei dati e definire un piano di ripristino prima di sfruttare la replica dei dati.

- Prima di attivare la SVM di destinazione di SnapMirror, interrompere tutti i trasferimenti pianificati di SnapMirror, interrompere tutti i trasferimenti in corso di SnapMirror, interrompere la relazione di replica, arrestare la SVM di origine e avviare la SVM di destinazione di SnapMirror.
- Astra Trident non rileva automaticamente gli errori SVM. Pertanto, in caso di errore, l'amministratore deve eseguire `tridentctl backend update` Comando per attivare il failover di Trident sul nuovo backend.

Di seguito viene riportata una panoramica delle fasi di installazione di SVM:

- Impostare il peering tra il cluster di origine e di destinazione e SVM.
- Creare la SVM di destinazione utilizzando `-subtype dp-destination` opzione.
- Creare una pianificazione dei processi di replica per garantire che la replica avvenga agli intervalli richiesti.
- Creare una replica SnapMirror dalla SVM di destinazione alla SVM di origine utilizzando `-identity -preserve true` Opzione per garantire che le configurazioni SVM di origine e le interfacce SVM di origine vengano copiate nella destinazione. Dalla SVM di destinazione, inizializzare la relazione di replica di SnapMirror SVM.



Workflow di disaster recovery per Trident

Astra Trident 19.07 e versioni successive utilizzano i CRD Kubernetes per memorizzare e gestire il proprio stato. Utilizza i cluster Kubernetes `etcd` per memorizzare i metadati. Supponiamo che i Kubernetes `etcd` I file di dati e i certificati vengono memorizzati su NetApp FlexVolume. Questo FlexVolume risiede in una SVM, che ha una relazione SnapMirror SVM-DR con una SVM di destinazione nel sito secondario.

I seguenti passaggi descrivono come ripristinare un singolo cluster Kubernetes master con Astra Trident in caso di disastro:

1. In caso di errore della SVM di origine, attivare la SVM di destinazione di SnapMirror. A tale scopo, è necessario interrompere i trasferimenti pianificati di SnapMirror, interrompere i trasferimenti in corso di SnapMirror, interrompere la relazione di replica, arrestare la SVM di origine e avviare la SVM di

destinazione.

2. Dalla SVM di destinazione, montare il volume che contiene Kubernetes `etcd` file di dati e certificati sull'host che verrà configurato come nodo master.
3. Copiare tutti i certificati richiesti relativi al cluster Kubernetes in `/etc/kubernetes/pki` e l'etcd `member` file sotto `/var/lib/etcd`.
4. Creare un cluster Kubernetes utilizzando `kubeadm init` con il `--ignore-preflight -errors=DirAvailable—var-lib-etcd` allarme. I nomi host utilizzati per i nodi Kubernetes devono essere gli stessi del cluster Kubernetes di origine.
5. Eseguire `kubectl get crd` Comando per verificare se tutte le risorse personalizzate di Trident sono state create e recuperare gli oggetti Trident per verificare che tutti i dati siano disponibili.
6. Aggiornare tutti i backend richiesti per riflettere il nuovo nome SVM di destinazione eseguendo il `./tridentctl update backend <backend-name> -f <backend-json-file> -n <namespace>` comando.

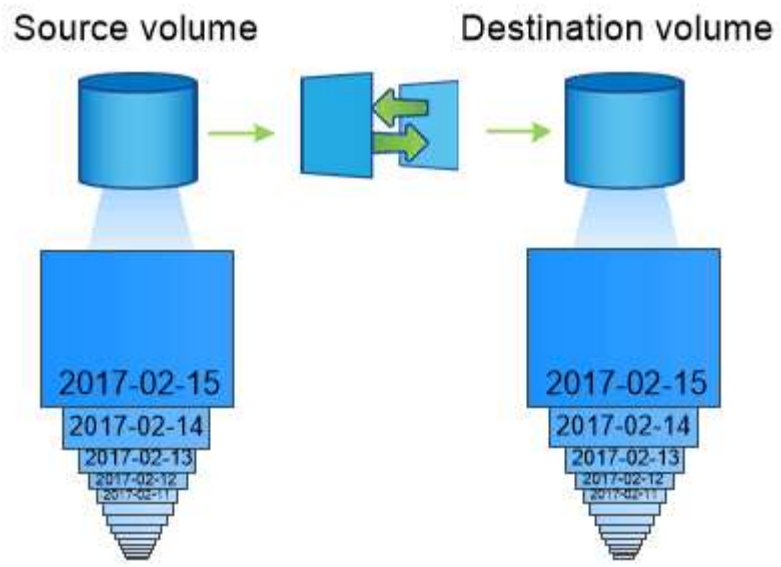
> Per i volumi persistenti dell'applicazione, quando viene attivata la SVM di destinazione, tutti i volumi forniti da Trident iniziano a servire i dati. Dopo aver configurato il cluster Kubernetes sul lato di destinazione seguendo i passaggi descritti in precedenza, vengono avviate tutte le implementazioni e i pod e le applicazioni containerizzate devono essere eseguite senza problemi.

Replica del volume SnapMirror

La replica dei volumi SnapMirror di ONTAP è una funzionalità di disaster recovery che consente il failover verso lo storage di destinazione dallo storage primario a livello di volume. SnapMirror crea una replica di volume o un mirror dello storage primario sullo storage secondario sincronizzando gli snapshot.

Di seguito viene riportata una panoramica dei passaggi per la configurazione della replica del volume di ONTAP SnapMirror:

- Impostare il peering tra i cluster in cui risiedono i volumi e le SVM che servono i dati dei volumi.
- Creare un criterio SnapMirror che controlli il comportamento della relazione e specifichi gli attributi di configurazione per tale relazione.
- Creare una relazione SnapMirror tra il volume di destinazione e il volume di origine utilizzando[`snapmirror create` Command^] e assegnare il criterio SnapMirror appropriato.
- Una volta creata la relazione SnapMirror, inizializzarla in modo da completare un trasferimento di riferimento dal volume di origine al volume di destinazione.



Workflow di disaster recovery del volume SnapMirror per Trident

I seguenti passaggi descrivono come ripristinare un singolo cluster Kubernetes master con Astra Trident.

1. In caso di disastro, interrompere tutti i trasferimenti SnapMirror pianificati e interrompere tutti i trasferimenti SnapMirror in corso. Interrompere la relazione di replica tra i volumi di destinazione e di origine in modo che il volume di destinazione diventi di lettura/scrittura.
2. Dalla SVM di destinazione, montare il volume che contiene Kubernetes `etcd` file di dati e certificati sull'host, che verrà impostato come nodo master.
3. Copiare tutti i certificati richiesti relativi al cluster Kubernetes in `/etc/kubernetes/pki` e l'etcd `member` file sotto `/var/lib/etcd`.
4. Creare un cluster Kubernetes eseguendo `kubeadm init` con il `--ignore-preflight -errors=DirAvailable—var-lib-etcd` allarme. I nomi host devono essere gli stessi del cluster Kubernetes di origine.
5. Eseguire `kubectl get crd` Per verificare se tutte le risorse personalizzate di Trident sono state create e recuperare gli oggetti Trident per assicurarsi che tutti i dati siano disponibili.
6. Ripulire i backend precedenti e creare nuovi backend su Trident. Specificare la nuova LIF di gestione e dati, il nuovo nome SVM e la password della SVM di destinazione.

Workflow di disaster recovery per volumi persistenti delle applicazioni

I seguenti passaggi descrivono come rendere disponibili i volumi di destinazione di SnapMirror per i carichi di lavoro containerizzati in caso di disastro:

1. Interrompere tutti i trasferimenti SnapMirror pianificati e interrompere tutti i trasferimenti SnapMirror in corso. Interrompere la relazione di replica tra il volume di destinazione e quello di origine in modo che il volume di destinazione diventi di lettura/scrittura. Ripulire le implementazioni che consumavano PVC legato ai volumi sulla SVM di origine.
2. Dopo aver configurato il cluster Kubernetes sul lato di destinazione seguendo le procedure descritte in precedenza, ripulire le implementazioni, PVC e PV, dal cluster Kubernetes.
3. Creare nuovi backend su Trident specificando la nuova LIF di gestione e dati, il nuovo nome SVM e la password della SVM di destinazione.

4. Importare i volumi richiesti come PV associato a un nuovo PVC utilizzando la funzione di importazione Trident.
5. Ridistribuire le implementazioni applicative con i PVC appena creati.

Ripristinare i dati utilizzando le snapshot Element

Eseguire il backup dei dati su un volume Element impostando una pianificazione di snapshot per il volume e garantendo che le snapshot vengano eseguite agli intervalli richiesti. È necessario impostare la pianificazione dello snapshot utilizzando l'interfaccia utente o le API di Element. Attualmente, non è possibile impostare una pianificazione snapshot su un volume tramite `solidfire-san` driver.

In caso di danneggiamento dei dati, è possibile scegliere uno snapshot specifico e eseguire il rollback del volume nello snapshot manualmente utilizzando l'interfaccia utente o le API Element. In questo modo vengono ripristinate le modifiche apportate al volume dalla creazione dello snapshot.

Sicurezza

Utilizzare i consigli elencati di seguito per assicurarsi che l'installazione di Astra Trident sia sicura.

Eseguire Astra Trident nel proprio namespace

È importante impedire ad applicazioni, amministratori di applicazioni, utenti e applicazioni di gestione di accedere alle definizioni degli oggetti di Astra Trident o ai pod per garantire uno storage affidabile e bloccare potenziali attività dannose.

Per separare le altre applicazioni e gli utenti da Astra Trident, installare sempre Astra Trident nel proprio spazio dei nomi Kubernetes (`trident`). L'inserimento di Astra Trident nel proprio spazio dei nomi garantisce che solo il personale amministrativo di Kubernetes abbia accesso al pod Astra Trident e agli artefatti (come i segreti di backend e CHAP, se applicabili) memorizzati negli oggetti CRD con spazio dei nomi. È necessario garantire che solo gli amministratori possano accedere allo spazio dei nomi Astra Trident e quindi a. `tridentctl` applicazione.

Utilizza l'autenticazione CHAP con i backend SAN ONTAP

Astra Trident supporta l'autenticazione basata su CHAP per i carichi di lavoro SAN ONTAP (utilizzando il `ontap-san` e. `ontap-san-economy` driver). NetApp consiglia di utilizzare CHAP bidirezionale con Astra Trident per l'autenticazione tra un host e il backend dello storage.

Per i backend ONTAP che utilizzano i driver di storage SAN, Astra Trident può configurare CHAP bidirezionale e gestire i nomi utente e i segreti CHAP attraverso `tridentctl`. Vedere "qui" Per capire come Astra Trident configura CHAP sui backend ONTAP.

> Il supporto CHAP per i backend ONTAP è disponibile con Trident 20.04 e versioni successive.

Utilizza l'autenticazione CHAP con backend NetApp HCI e SolidFire

NetApp consiglia di implementare CHAP bidirezionale per garantire l'autenticazione tra un host e i backend NetApp HCI e SolidFire. Astra Trident utilizza un oggetto segreto che include due password CHAP per tenant. Quando Trident viene installato come provider CSI, gestisce i segreti CHAP e li memorizza in un `tridentvolume` Oggetto CR per il rispettivo PV. Quando si crea un PV, CSI Astra Trident utilizza i segreti CHAP per avviare una sessione iSCSI e comunicare con il sistema NetApp HCI e SolidFire su CHAP.

I volumi creati da CSI Trident non sono associati a alcun gruppo di accesso al volume.

Nel frontend non CSI, l'attacco di volumi come dispositivi sui nodi di lavoro viene gestito da Kubernetes. Dopo la creazione del volume, Astra Trident effettua una chiamata API al sistema NetApp HCI/SolidFire per recuperare i segreti se il segreto per quel tenant non esiste già. Astra Trident poi passa i segreti su Kubernetes. Il kubelet situato su ciascun nodo accede ai segreti tramite l'API Kubernetes e li utilizza per eseguire/abilitare CHAP tra ciascun nodo che accede al volume e il sistema NetApp HCI/SolidFire in cui si trovano i volumi.

Riferimento

Porte Astra Trident

Scopri di più sulle porte su cui Astra Trident comunica.

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 il tempo di installazione utilizzando `--probe-port` allarme. È importante assicurarsi che questa porta non venga utilizzata da un altro processo sui nodi di lavoro.

API REST di Astra Trident

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

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

Per una maggiore sicurezza, Astra Trident's. `REST API` è limitato all'host locale per impostazione predefinita quando viene eseguito all'interno di un pod. Per modificare questo comportamento, devi impostare Astra Trident's. `-address` argomento nella configurazione del pod.

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 la creazione dell'oggetto. Per la specifica di ciascun tipo di oggetto, vedere il `tridentctl` comandi e opzioni.
- 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.

`DELETE`

`DELETE <trident-address>/trident/v1/<object-type>/<object-name>`: Elimina la risorsa indicata.

> I volumi associati ai backend o alle classi di storage continueranno ad esistere; questi devono essere cancellati separatamente. Per ulteriori informazioni, consulta il `tridentctl` comandi e opzioni.

Per esempi di come vengono chiamate queste API, passare il debug (`-d`). Per ulteriori informazioni, consulta il `tridentctl` comandi e opzioni.

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>`: Consente di impostare il livello di registrazione (debug, info, warn, error, fatale). Il valore predefinito è INFO.

Kubernetes

- `-k8s_pod`: Utilizzare questa opzione o. `-k8s_api_server` Per abilitare 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>`: 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. Ciò consente a Trident di essere implementato all'esterno di un pod; tuttavia, supporta solo connessioni non sicure al server API. Per una connessione sicura, implementare Trident in un pod con `-k8s_pod` opzione.
- `-k8s_config_path <file>`Obbligatorio; specificare questo percorso per un file KubeConfig.

Docker

- `-volume_driver <name>`: Nome del driver utilizzato durante la registrazione del plugin Docker. L'impostazione predefinita è `netapp`.
- `-driver_port <port-number>`: Ascoltare su questa porta piuttosto che su un socket di dominio UNIX.
- `-config <file>`Obbligatorio; specificare questo percorso per un file di configurazione back-end.

RIPOSO

- `-address <ip-or-host>`: Specifica l'indirizzo su cui il server REST di Trident deve essere in ascolto.

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 l'interfaccia REST accessibile 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 su cui il server REST di Trident deve essere in ascolto. Il valore predefinito è 8000.
- `-rest`: Attiva l'interfaccia REST. L'impostazione predefinita è true.

Prodotti NetApp integrati con Kubernetes

Il portfolio di prodotti storage NetApp si integra con molti aspetti diversi di un cluster Kubernetes, fornendo funzionalità avanzate di gestione dei dati che migliorano funzionalità, funzionalità, performance e disponibilità dell'implementazione di Kubernetes.

Astra

"Astra" Semplifica la gestione, la protezione e lo spostamento dei carichi di lavoro containerizzati ricchi di dati eseguiti su Kubernetes all'interno e tra cloud pubblici e on-premise. Astra fornisce e fornisce storage container persistente utilizzando Trident del comprovato e esteso portfolio di storage NetApp nel cloud pubblico e on-premise. Offre inoltre una serie completa di funzionalità avanzate di gestione dei dati applicative, come snapshot, backup e ripristino, log di attività e cloning attivo per la protezione dei dati, disaster recovery/data, audit dei dati e casi di utilizzo della migrazione per i carichi di lavoro Kubernetes.

ONTAP

ONTAP è il sistema operativo per lo storage unificato multiprotocollo di NetApp che offre funzionalità avanzate di gestione dei dati per qualsiasi applicazione. I sistemi ONTAP sono dotati di configurazioni all-flash, ibride o all-HDD e offrono diversi modelli di implementazione, tra cui hardware progettato (FAS e AFF), white-box (ONTAP Select) e solo cloud (Cloud Volumes ONTAP).

> Trident supporta tutti i modelli di implementazione ONTAP sopra menzionati.

Cloud Volumes ONTAP

"Cloud Volumes ONTAP" È un'appliance di storage solo software che esegue il software di gestione dei dati ONTAP nel cloud. Puoi utilizzare Cloud Volumes ONTAP per carichi di lavoro di produzione, disaster recovery, DevOps, condivisioni di file e gestione del database. Estende lo storage Enterprise al cloud offrendo efficienze dello storage, alta disponibilità, replica dei dati, tiering dei dati e coerenza applicativa.

Amazon FSX per NetApp ONTAP

"Amazon FSX per NetApp ONTAP" È un servizio AWS completamente gestito che consente ai clienti di lanciare ed eseguire file system basati sul sistema operativo per lo storage ONTAP di NetApp. FSX per ONTAP consente ai clienti di sfruttare le funzionalità, le performance e le funzionalità amministrative di NetApp con cui hanno familiarità, sfruttando al contempo la semplicità, l'agilità, la sicurezza e la scalabilità dell'archiviazione dei dati su AWS. FSX per ONTAP supporta molte delle funzionalità del file system e delle API di amministrazione di ONTAP.

Software Element

"Elemento" consente all'amministratore dello storage di consolidare i carichi di lavoro garantendo le performance e consentendo un footprint dello storage semplificato e ottimizzato. Insieme a un'API per consentire l'automazione di tutti gli aspetti della gestione dello storage, Element consente agli amministratori dello storage di fare di più con meno sforzo.

NetApp HCI

"NetApp HCI" semplifica la gestione e la scalabilità del data center automatizzando le attività di routine e consentendo agli amministratori dell'infrastruttura di concentrarsi su funzioni più importanti.

NetApp HCI è completamente supportato da Trident. Trident è in grado di eseguire il provisioning e la gestione dei dispositivi di storage per le applicazioni containerizzate direttamente sulla piattaforma di storage NetApp HCI sottostante.

Azure NetApp Files

"Azure NetApp Files" È un servizio di condivisione file Azure di livello Enterprise, basato su NetApp. Puoi eseguire i carichi di lavoro basati su file più esigenti in Azure in modo nativo, con le performance e la gestione completa dei dati che ti aspetti da NetApp.

Cloud Volumes Service per Google Cloud

"NetApp Cloud Volumes Service per Google Cloud" È un file service nativo nel cloud che fornisce volumi NAS su NFS e SMB con performance all-flash. Questo servizio consente l'esecuzione di qualsiasi workload, incluse le applicazioni legacy, nel cloud GCP. Offre un servizio completamente gestito che offre performance elevate e costanti, cloning istantaneo, protezione dei dati e accesso sicuro alle istanze di Google Compute Engine (GCE).

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 `PersistentVolumeClaim` richiesta di un nuovo `PersistentVolume` Di una dimensione particolare da un Kubernetes `StorageClass` precedentemente configurato dall'amministratore.
2. Kubernetes `StorageClass` Identifica Trident come provider e include parametri che indicano a Trident come eseguire il provisioning di un volume per la classe richiesta.
3. Trident si guarda da solo `StorageClass` 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 su un backend corrispondente e crea due oggetti: A. `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 il `PersistentVolumeClaim` al nuovo `PersistentVolume`. Pod che includono `PersistentVolumeClaim` Montare il PersistentVolume su qualsiasi host su cui viene eseguito.
6. Un utente crea un `VolumeSnapshot` Di un PVC esistente, utilizzando un `VolumeSnapshotClass` Questo indica Trident.
7. Trident identifica il volume associato al PVC e crea un'istantanea del volume sul backend. Inoltre, crea un `VolumeSnapshotContent` Che indica a Kubernetes come identificare lo snapshot.
8. Un utente può creare un `PersistentVolumeClaim` utilizzo di `VolumeSnapshot` come fonte.
9. Trident identifica lo snapshot richiesto ed esegue la stessa serie di passaggi necessari per la creazione di `PersistentVolume` e a. `Volume`.

Per ulteriori informazioni sugli oggetti Kubernetes, si consiglia di leggere il "Volumi persistenti" Della documentazione Kubernetes.

Kubernetes `PersistentVolumeClaim` oggetti

Un Kubernetes `PersistentVolumeClaim` Object è una richiesta di storage effettuata 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, eseries-iscsi, 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 di `Delete` Recuperare la policy, 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` Policy, Trident lo ignora e presuppone che l'amministratore lo pulisca da Kubernetes e dal backend, consentendo 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. Dai un'occhiata a. `On-Demand Volume Snapshots` per vedere come funziona.

Trident fornisce anche `cloneFromPVC` e. `splitOnClone` annotazioni per la creazione di cloni. È possibile utilizzare queste annotazioni per clonare un PVC senza utilizzare l'implementazione CSI (su Kubernetes 1.13 e versioni precedenti) o se la release di Kubernetes non supporta le snapshot dei volumi beta (Kubernetes 1.16 e versioni precedenti). Tenere presente che Trident 19.10 supporta il workflow CSI per la clonazione da PVC.

> È possibile utilizzare `cloneFromPVC` e. `splitOnClone` Annotazioni con CSI Trident e con il frontend tradizionale non 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, ad esempio `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 `ontap-nas` e. `ontap-san` Driver, potrebbe essere consigliabile impostare l'annotazione PVC `trident.netapp.io/splitOnClone` in combinazione con `trident.netapp.io/cloneFromPVC`. Con `trident.netapp.io/splitOnClone` impostare su `true`, Trident suddivide il volume clonato dal volume padre e, di conseguenza, disaccadeva completamente il ciclo di vita del volume clonato dal volume padre a scapito di una certa efficienza dello storage. Non impostato `trident.netapp.io/splitOnClone` o impostarlo su `false` si ottiene un consumo di spazio ridotto sul backend a scapito della creazione di dipendenze tra i volumi padre e clone, in modo che il volume padre non possa essere cancellato a meno che il clone non venga cancellato 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.

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

Kubernetes `PersistentVolume` oggetti

Un Kubernetes `PersistentVolume` Object rappresenta un elemento di storage che viene reso disponibile per il cluster Kubernetes. Ha un ciclo di vita indipendente dal pod che lo utilizza.

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

Quando si crea un PVC che si riferisce a un Trident-based `StorageClass`, Trident esegue il provisioning di un nuovo volume utilizzando la classe di storage 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.

Kubernetes `StorageClass` oggetti

Kubernetes `StorageClass` gli oggetti sono specificati in base al nome `PersistentVolumeClaims` per eseguire 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 Kubernetes `StorageClass` L'oggetto che utilizza Trident è simile al seguente:

```
apiVersion: storage.k8s.io/v1beta1
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

Attributo	Tipo	Obbligatorio	Descrizione
AddtionalStoragePools	map[string]StringList	no	Mappatura dei nomi backend agli elenchi di pool di storage all'interno di
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 e eseries-iscsi; thin: all ONTAP e solidfire-san
BackendType	stringa	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, eseries-iscsi, 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

Attributo	Tipo	Valori	Offerta	Richiesta	Supportato da
crittografia	bool	vero, falso	Il pool supporta volumi crittografati	Volume con crittografia attivata	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
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 soli per modellare le qualità dello storage necessarie per soddisfare le esigenze di una particolare classe. Trident rileva e seleziona automaticamente i pool di storage che corrispondono a *tutti* di `attributes` specificato dall'utente.

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

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

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

È possibile utilizzare `excludeStoragePools` Parametro per filtrare il set di pool che Trident utilizza per il provisioning. L'utilizzo di questo parametro consente di rimuovere i pool corrispondenti.

In `storagePools` e. `additionalStoragePools` parametri, ogni voce assume la forma `<backend>:<storagePoolList>`, dove `<storagePoolList>` è un elenco separato da virgole di pool di storage per il backend specificato. Ad esempio, un valore per `additionalStoragePools` potrebbe sembrare `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. È possibile utilizzare `tridentctl get backend` per ottenere l'elenco dei backend e dei relativi pool.

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, ecc.	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, eseries-iscsi	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	1.17+

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

  Per i driver di storage NFS, esiste già un filesystem come parte dell'esportazione NFS. Per l'utilizzo `fsGroup` la classe di storage deve ancora specificare un `fsType`. È possibile impostarlo su `nfs` o qualsiasi valore non nullo.
- Vedere "Espandere i volumi" per ulteriori dettagli sull'espansione dei volumi.
- Il bundle del programma di installazione Trident fornisce diverse definizioni di classi di storage 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.

Kubernetes `VolumeSnapshotClass` oggetti

Kubernetes `VolumeSnapshotClass` gli oggetti 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.

R `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/v1beta1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

Il `driver` Specifica a Kubernetes che richiede snapshot di volume di `csi-snapclass` Le classi sono gestite da Trident. Il `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 storage vengono rimossi quando viene eliminata una snapshot. In alternativa, impostarla su `Retain` significa che `VolumeSnapshotContent` e lo snapshot fisico viene conservato.

Kubernetes `VolumeSnapshot` oggetti

Un Kubernetes `VolumeSnapshot` object è una richiesta per creare uno 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 back-end ed espone lo snapshot creando un unico `VolumeSnapshotContent` oggetto. 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.

Kubernetes `VolumeSnapshotContent` oggetti

Un Kubernetes `VolumeSnapshotContent` object rappresenta uno snapshot preso da un volume già sottoposto a provisioning. È analogo a a. `PersistentVolume` e indica uno snapshot con provisioning sul cluster di storage. Simile a. `PersistentVolumeClaim` e. `PersistentVolume` oggetti, quando viene creata una snapshot, il `VolumeSnapshotContent` l'oggetto mantiene un mapping uno a uno a `VolumeSnapshot` oggetto, che aveva richiesto la creazione dello snapshot.

> Trident crea `VolumeSnapshotContent` E li registra automaticamente con il cluster Kubernetes in base ai volumi forniti. Non ci si aspetta di gestirli da soli.

Il `VolumeSnapshotContent` oggetto contiene dettagli che identificano in modo univoco lo snapshot, ad esempio `snapshotHandle`. Questo `snapshotHandle` È una combinazione univoca del nome del PV e del nome del `VolumeSnapshotContent` oggetto.

Quando arriva una richiesta di snapshot, Trident crea lo snapshot sul back-end. Una volta creata la snapshot, Trident configura una `VolumeSnapshotContent` E quindi espone lo snapshot all'API Kubernetes.

Kubernetes `CustomResourceDefinition` oggetti

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.

A partire dalla versione 19.07, Trident utilizza una serie di `CustomResourceDefinition` Oggetti per preservare l'identità degli oggetti Trident, come backend Trident, classi di storage Trident e 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 viene creato un backend utilizzando `tridentctl`, un corrispondente `tridentbackends` L'oggetto CRD viene creato per l'utilizzo da parte di 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 esegue l'aggiornamento da una versione precedente di Trident (quella utilizzata `etcd` Per mantenere lo stato), il programma di installazione di Trident esegue la migrazione dei dati da `etcd` Archiviazione dei dati Key-Value e creazione degli oggetti CRD corrispondenti.
- Quando si disinstalla Trident utilizzando `tridentctl uninstall` Comando, i pod Trident vengono cancellati ma i CRD creati non vengono ripuliti. Vedere "Disinstallare Trident" Per capire come Trident può essere completamente rimosso e riconfigurato da zero.

Trident `StorageClass` oggetti

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

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

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 di Kubernetes, ci aspettiamo che vengano create al momento della registrazione dei nuovi Kubernetes `StorageClass` oggetti.

Oggetti backend 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 Kubernetes `StorageClass` oggetto.

Per ulteriori informazioni sulla creazione di questi oggetti, vedere Configurazione back-end.

Trident `StoragePool` oggetti

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.

Trident `Volume` oggetti

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. Prima di tutto, prepende il prefisso di storage (predefinito) `trident` o il prefisso nella configurazione back-end) al nome del volume, con conseguente nome del modulo `<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 le configurazioni dei volumi per eseguire il provisioning diretto dei volumi utilizzando l'API REST, ma nelle implementazioni di Kubernetes ci aspettiamo che la maggior parte degli utenti utilizzi il Kubernetes standard `PersistentVolumeClaim` metodo. Trident crea automaticamente questo oggetto volume come parte del processo di provisioning.

Trident `Snapshot` oggetti

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 a.

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

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

comandi e opzioni tridentctl

Il "Pacchetto di installazione Trident" include un'utility della riga di comando, `tridentctl`, Che fornisce un accesso semplice ad Astra Trident. Kubernetes gli utenti con privilegi sufficienti possono utilizzarlo per installare Astra Trident e per interagire direttamente con esso per gestire lo spazio dei nomi che contiene il pod Astra Trident.

Per informazioni sull'utilizzo, eseguire `tridentctl --help`.

I comandi e le opzioni globali disponibili sono:

```
Usage:
  tridentctl [command]
```

Comandi disponibili:

- `create`: Aggiungere una risorsa ad Astra Trident.

- `delete`: Rimuovere una o più risorse da Astra Trident.
- `get`: Ottieni una o più risorse da Astra Trident.
- `help`: Guida su qualsiasi comando.
- `images`: Stampare una tabella delle immagini container di Astra Trident.
- `import`: Importare una risorsa esistente in Astra Trident.
- `install`: Installare Astra Trident.
- `logs`: Stampare i log da Astra Trident.
- `send`: Inviare una risorsa da Astra Trident.
- `uninstall`: Disinstallare Astra Trident.
- `update`: Modificare una risorsa in Astra Trident.
- `upgrade`: Aggiornare una risorsa in Astra Trident.
- `version`: Stampa la versione di Astra Trident.

Allarmi:

- `` `-d, --debug ``: Output di debug.
- `` `-h, --help ``: Guida per `tridentctl`.
- `` `-n, --namespace string ``: Namespace dell'implementazione di Astra Trident.
- `` `-o, --output string ``: Formato di output. Uno tra json|yaml|name|wide|ps (impostazione predefinita).
- `` `-s, --server string ``: Indirizzo/porta dell'interfaccia REST di Astra Trident.

> 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).

> 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).

`create`

È possibile utilizzare il `create` Comando per aggiungere una risorsa ad Astra Trident.

```
Usage:
  tridentctl create [option]
```

Opzione disponibile:
`backend`: Aggiungi un backend ad Astra Trident.

`delete`

È possibile eseguire `delete` Comando per rimuovere una o più risorse da Astra Trident.

```
Usage:
  tridentctl delete [option]
```

Opzioni disponibili:

- `backend`: Eliminare uno o più backend di storage da Astra Trident.
- `node`: Eliminare uno o più nodi CSI da Astra Trident.
- `snapshot`: Consente di eliminare una o più snapshot di volumi da Astra Trident.
- `storageclass`: Eliminare una o più classi di storage da Astra Trident.
- `volume`: Eliminare uno o più volumi di storage da Astra Trident.

`get`

È possibile eseguire `get` Comando per ottenere una o più risorse da Astra Trident.

```
Usage:
  tridentctl get [option]
```

Opzioni disponibili:

- `backend`: Ottieni uno o più backend di storage da Astra Trident.
- `snapshot`: Ottenere una o più istantanee da Astra Trident.
- `storageclass`: Ottieni una o più classi di storage da Astra Trident.
- `volume`: Procurarsi uno o più volumi da Astra Trident.

`images`

È possibile eseguire `images` Flag per stampare una tabella delle immagini container richieste da Astra Trident.

```
Usage:
  tridentctl images [flags]
```

Allarmi: * `-h, --help`: Help for images.
* -V, --k8s-version string`: Versione semantica del cluster Kubernetes.

`import volume`

È possibile eseguire `import volume` Comando per importare un volume esistente in Astra Trident.

```
Usage:
  tridentctl import volume <backendName> <volumeName> [flags]
```

Alias:
`volume, v`

Allarmi:

- `` `-f, --filename string`: Percorso al file PVC YAML o JSON.
- `` `-h, --help`: Guida per il volume.
- `` `--no-manage`: Crea solo PV/PVC. Non presupporre la gestione del ciclo di vita dei volumi.

`install`

È possibile eseguire `install` Flag per installare Astra Trident.

```
Usage:
  tridentctl install [flags]
```

Allarmi:

- `` `--autosupport-image string`: L'immagine container per il sistema di Telemetria AutoSupport (impostazione predefinita "netapp/Trident autosupport:20.07.0").
- ``--autosupport-proxy string`Indirizzo/porta di un proxy per l'invio di telemetria AutoSupport.
- `` `--csi`: Installare CSI Trident (override solo per Kubernetes 1.13, richiede feature gate).
- `` `--enable-node-prep`: Tentare di installare i pacchetti richiesti sui nodi.
- `` `--generate-custom-yaml`: Generare file YAML senza installare nulla.
- `` `-h, --help`: Guida all'installazione.
- `` `--http-request-timeout`: Consente di sovrascrivere il timeout della richiesta HTTP per l'API REST del controller Trident (valore predefinito 1 m30s).
- `` `--image-registry string`: L'indirizzo/porta di un registro di immagini interno.
- `` `--k8s-timeout duration`: Il timeout per tutte le operazioni Kubernetes (valore predefinito: 3 m0s).
- `` `--kubelet-dir string`: La posizione host dello stato interno di kubelet (default "/var/lib/kubelet").
- `` `--log-format string`: Il formato di registrazione Astra Trident (text, json) (default "text").
- `` `--pv string`: Il nome del PV legacy utilizzato da Astra Trident, garantisce che non esista (il "tridente" predefinito).
- `` `--pvc string`: Il nome del PVC legacy utilizzato da Astra Trident, garantisce che non esista (il "tridente" predefinito).
- `` `--silence-autosupport`: Non inviare pacchetti AutoSupport a NetApp automaticamente (valore predefinito vero).

- `--silent: Disattivare l'output MOST durante l'installazione.
- `--trident-image string: L'immagine Astra Trident da installare.
- `--use-custom-yaml: Utilizzare tutti i file YAML esistenti nella directory di installazione.
- `--use-ipv6: Utilizza IPv6 per la comunicazione di Astra Trident.

logs

È possibile eseguire `logs` Flag per stampare i log da Astra Trident.

```
Usage:
  tridentctl logs [flags]
```

Allarmi:

- `-a, --archive: Creare un archivio di supporto con tutti i log, se non diversamente specificato.
- `-h, --help: Guida per i log.
- `-l, --log string: Registro Astra Trident da visualizzare. Uno tra trident|auto|trident-operator|all (impostazione predefinita "auto").
- `--node string: Il nome del nodo Kubernetes da cui raccogliere i log dei pod dei nodi.
- `-p, --previous: Ottenere i log per l'istanza di container precedente, se esistente.
- `--sidecars: Ottenere i log per i contenitori del sidecar.

send

È possibile eseguire `send` Comando per inviare una risorsa da Astra Trident.

```
Usage:
  tridentctl send [option]
```

Opzione disponibile:
`autosupport`: Inviare un archivio AutoSupport a NetApp.

uninstall

È possibile eseguire `uninstall` Flag per disinstallare Astra Trident.

```
Usage:
  tridentctl uninstall [flags]
```

Allarmi: * `-h, --help`: Guida per la disinstallazione. * `--silent`: Disattivare l'output MOST durante la disinstallazione.

update

È possibile eseguire `update` Comandi per modificare una risorsa in Astra Trident.

```
Usage:
  tridentctl update [option]
```

Opzioni disponibili:
`backend`: Aggiornare un backend in Astra Trident.

upgrade

È possibile eseguire `upgrade` Comandi per aggiornare una risorsa in Astra Trident.

```
Usage:
tridentctl upgrade [option]
```

Opzione disponibile:
`volume`: Aggiornare uno o più volumi persistenti da NFS/iSCSI a CSI.

version

È possibile eseguire `version` contrassegni per stampare la versione di `tridentctl` E il servizio Running Trident.

```
Usage:
  tridentctl version [flags]
```

Allarmi: * `--client`: Solo versione client (non è richiesto alcun server). * `-h, --help`: Guida per la versione.

Versioni precedenti della documentazione

Se non si utilizza Astra Trident 22.01, è disponibile la documentazione relativa alle release precedenti.

- "Astra Trident 21.10"
- "Astra Trident 21.07"

Note legali

Le note legali forniscono l'accesso a dichiarazioni di copyright, marchi, brevetti e altro ancora.

Copyright

"https://www.netapp.com/company/legal/copyright/"

Marchi

NETAPP, il logo NETAPP e i marchi elencati nella pagina dei marchi NetApp sono marchi di NetApp, Inc. Altri nomi di società e prodotti potrebbero essere marchi dei rispettivi proprietari.

"https://www.netapp.com/company/legal/trademarks/"

Brevetti

Un elenco aggiornato dei brevetti di proprietà di NetApp è disponibile all'indirizzo:

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Direttiva sulla privacy

"https://www.netapp.com/company/legal/privacy-policy/"

Open source

È possibile consultare il copyright e le licenze di terze parti utilizzate nel software NetApp per Astra Trident nel file degli avvisi per ciascuna release all'indirizzo https://github.com/NetApp/trident/.