Documentazione di Astra Trident 24,02

Astra Trident

NetApp
January 14, 2026

This PDF was generated from https://docs.netapp.com/it-it/trident-2402/index.html on January 14, 2026.
Always check docs.netapp.com for the latest.

Sommario

Documentazione di Astra Trident 24,02 . . . 1
Note di rilascio . . . 2
Novità . . . 2
Novità del 24,02 . . . 2
Modifiche nel 23,10 . . . 2
Modifiche nel 23.07.1 . . . 3
Modifiche nel 23,07 . . . 3
Modifiche nel 23,04 . . . 4
Cambiamenti nel 23.01.1 . . . 5
Cambiamenti nel 23.01 . . . 6
Cambiamenti nel 22.10 . . . 6
Cambiamenti nel 22.07 . . . 8
Cambiamenti nel 22.04 . . . 9
Cambiamenti nel 22.01.1 . . . 9
Cambiamenti nel 22.01.0 . . . 9
Cambiamenti nel 21.10.1 . . . 10
Cambiamenti nel 21.10.0 . . . 11
Problemi noti . . . 11
Trova ulteriori informazioni . . . 12
Versioni precedenti della documentazione . . . 13
Inizia subito . . . 14
Scopri Astra Trident . . . 14
Scopri Astra Trident . . . 14
Architettura Astra Trident . . . 16
Concetti . . . 19
Avvio rapido di Astra Trident . . . 23
Quali sono le prossime novità? . . . 24
Requisiti . . . 24
Informazioni critiche su Astra Trident . . . 24
Frontend supportati (orchestratori) . . . 24
Back-end supportati (storage) . . . 25
Requisiti delle funzionalità . . . 25
Sistemi operativi host testati . . . 25
Configurazione dell'host . . . 26
Configurazione del sistema storage . . . 26
Porte Astra Trident . . . 26
Immagini container e corrispondenti versioni di Kubernetes . . . 26
Installare Astra Trident . . . 30
Installare utilizzando l'operatore Trident . . . 30
Installare usando tridentctl . . . 30
Utilizzare Astra Trident . . . 31
Preparare il nodo di lavoro . . . 31
Selezionare gli strumenti giusti . . . 31

Rilevamento del servizio del nodo . . . 31
Volumi NFS . . . 32
Volumi iSCSI . . . 32
Volumi NVMe/TCP . . . 36
Configurare e gestire i backend . . . 37
Configurare i backend . . . 37
Azure NetApp Files . . . 37
Configurare un Cloud Volumes Service per il backend di Google Cloud . . . 54
Configurare un backend NetApp HCI o SolidFire . . . 70
Driver SAN ONTAP . . . 76
Driver NAS ONTAP . . . 99
Amazon FSX per NetApp ONTAP . . . 128
Crea backend con kubectl . . . 146
Gestire i backend . . . 153
Creare e gestire classi di archiviazione . . . 162
Creare una classe di storage . . . 162
Gestire le classi di storage . . . 165
Provisioning e gestione dei volumi . . . 167
Provisioning di un volume . . . 167
Espandere i volumi . . . 171
Importa volumi . . . 179
Condividere un volume NFS tra spazi dei nomi . . . 186
Replica dei volumi con SnapMirror . . . 189
Utilizzare la topologia CSI . . . 205
Lavorare con le istantanee . . . 213
Gestisci e monitora Astra Trident . . . 222
Aggiorna Astra Trident . . . 222
Aggiorna Astra Trident . . . 222
Eseguire l'upgrade con l'operatore . . . 223
Upgrade con tridentctl . . . 228
Gestisci Astra Trident usando tridentctl . . . 228
Comandi e flag globali . . . 228
Opzioni di comando e flag . . . 230
Monitorare Astra Trident . . . 233
Panoramica . . . 234
Fase 1: Definire un target Prometheus . . . 234
Fase 2: Creazione di un ServiceMonitor Prometheus . . . 234
Fase 3: Eseguire una query sulle metriche di Trident con PromQL . . . 235
Scopri di più sulla telemetria Astra Trident AutoSupport . . . 236
Disattiva le metriche di Astra Trident . . . 237
Disinstallare Astra Trident . . . 237
Determinare il metodo di installazione originale . . . 238
Disinstallare un'installazione dell'operatore Trident . . . 238
Disinstallare un `tridentctl` installazione . . . 239
Astra Trident per Docker . . . 240

Prerequisiti per l'implementazione . . . 240
Verificare i requisiti . . . 240
Strumenti NVMe . . . 242
Implementare Astra Trident . . . 243
Metodo del plugin gestito da Docker (versione 1.13/17.03 e successive) . . . 243
Metodo tradizionale (versione 1.12 o precedente) . . . 245
Avviare Astra Trident all'avvio del sistema . . . 247
Aggiornare o disinstallare Astra Trident . . . 248
Eseguire l'upgrade . . . 248
Disinstallare . . . 249
Lavorare con i volumi . . . 249
Creare un volume . . . 250
Rimuovere un volume . . . 250
Clonare un volume . . . 250
Accesso ai volumi creati esternamente . . . 252
Opzioni di volume specifiche del driver . . . 252
Raccogliere i log . . . 258
Raccogliere i registri per la risoluzione dei problemi . . . 258
Suggerimenti generali per la risoluzione dei problemi . . . 258
Gestire più istanze di Astra Trident . . . 259
Procedura per il plug-in gestito da Docker (versione 1.13/17.03 o successiva) . . . 259
Procedura per la versione tradizionale (1.12 o precedente) . . . 259
Opzioni di configurazione dello storage . . . 260
Opzioni di configurazione globale . . . 260
Configurazione di ONTAP . . . 261
Configurazione del software Element . . . 268
Problemi noti e limitazioni . . . 270
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. . . . 270
I nomi dei volumi devono contenere almeno 2 caratteri. . . . 271
Docker Swarm ha alcuni comportamenti che impediscono ad Astra Trident di supportarlo con ogni combinazione di storage e driver. . . . 271
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. . . . 271
Best practice e consigli . . . 272
Implementazione . . . 272
Eseguire l'implementazione in uno spazio dei nomi dedicato . . . 272
Utilizza quote e limiti di intervallo per controllare il consumo dello storage . . . 272
Configurazione dello storage . . . 272
Panoramica della piattaforma . . . 272
Best practice per ONTAP e Cloud Volumes ONTAP . . . 272
Best practice di SolidFire . . . 277
Dove trovare ulteriori informazioni? . . . 279
Integrare Astra Trident . . . 279

Selezione e implementazione dei driver . . . 279
Design di classe storage . . . 283
Progettazione di un pool virtuale . . . 284
Operazioni di volume . . . 285
Implementare i servizi OpenShift . . . 286
Servizio di metriche . . . 289
Protezione dei dati e disaster recovery . . . 290
Replica e recovery di Astra Trident . . . 290
Replica e recovery di SVM . . . 291
Replica e recovery dei volumi . . . 292
Protezione dei dati Snapshot . . . 292
Replica dell'applicazione Astra Control Center . . . 292
Sicurezza . . . 292
Sicurezza . . . 292
Linux Unified Key Setup (LUKS) . . . 294
Configurare la crittografia Kerberos in-flight . . . 298
Conoscenza e supporto . . . 307
Domande frequenti . . . 307
Domande generali . . . 307
Installare e utilizzare Astra Trident su un cluster Kubernetes . . . 307
Risoluzione dei problemi e supporto . . . 308
Aggiorna Astra Trident . . . 309
Gestione di back-end e volumi . . . 310
Risoluzione dei problemi . . . 314
Risoluzione dei problemi generali . . . 314
Implementazione Trident non riuscita utilizzando l'operatore . . . 316
Implementazione Trident non riuscita utilizzando `tridentctl` . . . 318
Rimuovere completamente Astra Trident e i CRD . . . 318
Guasto durante l'unstadiazione del nodo NVMe con namespace di blocchi raw RWX o Kubernetes 1,26 . . . 319
Supporto . . . 320
Ciclo di vita del supporto di Astra Trident . . . 320
Supporto autonomo . . . 321
Sostegno della community . . . 321
Assistenza tecnica NetApp . . . 321
Per ulteriori informazioni . . . 321
Riferimento . . . 322
Porte Astra Trident . . . 322
Porte Astra Trident . . . 322
API REST di Astra Trident . . . 322
Quando utilizzare l'API REST . . . 322
Utilizzo dell'API REST . . . 322
Opzioni della riga di comando . . . 323
Registrazione . . . 323
Kubernetes . . . 323

Docker . . . 324
RIPOSO . . . 324
Kubernetes e Trident Objects . . . 324
In che modo gli oggetti interagiscono tra loro? . . . 324
Kubernetes `PersistentVolumeClaim` oggetti . . . 325
Kubernetes `PersistentVolume` oggetti . . . 327
Kubernetes `StorageClass` oggetti . . . 327
Kubernetes `VolumeSnapshotClass` oggetti . . . 331
Kubernetes `VolumeSnapshot` oggetti . . . 331
Kubernetes `VolumeSnapshotContent` oggetti . . . 331
Kubernetes `CustomResourceDefinition` oggetti . . . 332
Astra Trident `StorageClass` oggetti . . . 332
Oggetti di backend Astra Trident . . . 333
Astra Trident `StoragePool` oggetti . . . 333
Astra Trident `Volume` oggetti . . . 333
Astra Trident `Snapshot` oggetti . . . 334
Astra Trident `ResourceQuota` oggetto . . . 335
Pod Security Standards (PSS) e Security Context Constraints (SCC) . . . 336
Contesto di sicurezza Kubernetes obbligatorio e campi correlati . . . 337
Standard di sicurezza Pod (PSS) . . . 337
Policy di sicurezza Pod (PSP) . . . 338
SCC (Security Context Constraints) . . . 339
Note legali . . . 342
Copyright . . . 342
Marchi . . . 342
Brevetti . . . 342
Direttiva sulla privacy . . . 342
Open source . . . 342

Documentazione di Astra Trident 24,02

Note di rilascio

Novità

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 24,02

Miglioramenti

- Aggiunto supporto per Cloud Identity.
  - AKS con ANF - Azure workload Identity verrà utilizzato come Cloud Identity.
  - EKS con FSxN - il ruolo AWS IAM verrà utilizzato come identità Cloud.
- Aggiunto il supporto per installare Astra Trident come add-on sul cluster EKS dalla console EKS.
- È stata aggiunta la possibilità di configurare e disattivare la riparazione automatica iSCSI ("Problema n. 864").
- È stata aggiunta la personalità FSX ai driver ONTAP per consentire l'integrazione con AWS IAM e SecretsManager e per consentire ad Astra Trident di eliminare i volumi FSX con i backup ("Problema n. 453").

Kubernetes

- Aggiunto supporto per Kubernetes 1,29.

Correzioni

- Messaggi di avviso ACP fissi, quando ACP non è abilitato ("Problema n. 866").
- È stato aggiunto un ritardo di 10 secondi prima di eseguire una suddivisione dei cloni durante l'eliminazione dello snapshot per i driver ONTAP, quando un clone è associato allo snapshot.

Dipendenze

- Rimosso il framework degli attestati in-toto dai manifesti di immagini multipiattaforma.

Modifiche nel 23,10

Correzioni

- Espansione del volume fisso se la nuova dimensione richiesta è inferiore alle dimensioni del volume totale per i driver di storage ontap-nas e ontap-nas-flexgroup ("Problema n. 834").
- Dimensioni fisse del volume per visualizzare solo le dimensioni utilizzabili del volume durante l'importazione per i driver di storage ontap-nas e ontap-nas-flexgroup ("Problema n. 722").

- Conversione fissa del nome FlexVol per ONTAP-NAS-Economy.
- Risolto il problema di inizializzazione di Astra Trident su un nodo Windows quando il nodo viene riavviato.

Miglioramenti

Kubernetes

Aggiunto supporto per Kubernetes 1,28.

Astra Trident

- Aggiunto supporto per l'utilizzo di Azure Managed Identity (AMI) con driver di storage Azure-netapp-Files.
- Aggiunto supporto per NVMe su TCP per il driver ONTAP-SAN.
- Aggiunta la possibilità di sospendere il provisioning di un volume quando il backend è impostato sullo stato sospeso dall'utente ("Problema n. 558").

Funzionalità avanzate disponibili in Astra Control

Con Astra Trident 23,10, un nuovo componente software chiamato Astra Control Provivisioner è disponibile per gli utenti con licenza Astra Control. Questo provisioner fornisce l'accesso a un superset di funzionalità avanzate di gestione e provisioning dello storage, oltre a quelle supportate da Astra Trident. Per la release 23,10, queste funzionalità includono:

- Funzionalità di backup e ripristino per le applicazioni con backend di storage con garanzia di driver basati sulla convenienza di ontap-nas
- Maggiore sicurezza backend dello storage con crittografia Kerberos 5
- Recovery di dati con snapshot
- Miglioramenti di SnapMirror

"Scopri di più su Astra Control Provisioner."

Modifiche nel 23.07.1

Kubernetes: eliminazione di daemonset fissa per supportare aggiornamenti senza downtime (."Problema n. 740").

Modifiche nel 23,07

Correzioni

Kubernetes

- Risolto l'aggiornamento Trident per ignorare i vecchi pod bloccati in stato di terminazione ("Problema n. 740").
- Aggiunta tolleranza alla definizione "versione-pod-tridente-transitorio" ("Problema n. 795").

Astra Trident

- Sono state risolte le richieste ZAPI ONTAP per garantire che i numeri di serie LUN vengano interrogati quando si ottengono gli attributi LUN per identificare e correggere i dispositivi iSCSI fantasma durante le operazioni di staging dei nodi.

- Correzione della gestione degli errori nel codice del driver di archiviazione ("Problema n. 816").
- Risolto il ridimensionamento delle quote quando si utilizzano i driver ONTAP con use-REST=true.
- Creazione di cloni di LUN fissi in ontap-san-economy.
- Ripristina campo informazioni di pubblicazione da `rawDevicePath` a. `devicePath`; aggiunta della logica per popolare e recuperare (in alcuni casi) `devicePath` campo.

Miglioramenti

Kubernetes

- Aggiunto supporto per l'importazione di snapshot pre-sottoposte a provisioning.
- Distribuzione ridotta al minimo e permessi linux daemesort ("Problema n. 817").

Astra Trident

- Non è più necessario specificare il campo dello stato per volumi e snapshot "online".
- Aggiorna lo stato backend se il backend ONTAP è offline ("Numeri 801", "N. 543").
- Il numero di serie LUN viene sempre recuperato e pubblicato durante il flusso di lavoro ControllerVolumePublish.
- Aggiunta logica aggiuntiva per verificare il numero di serie e le dimensioni del dispositivo multipath iSCSI.
- Verifica aggiuntiva dei volumi iSCSI per assicurare che il dispositivo multipath corretto non venga messo in fase.

Miglioramento sperimentale

Aggiunto il supporto dell'anteprima tecnica per NVMe su TCP per il driver ONTAP-SAN.

Documentazione

Sono stati apportati molti miglioramenti a livello organizzativo e di formattazione.

Dipendenze

Kubernetes

- Supporto rimosso per istantanee v1beta1.
- Rimosso il supporto per volumi e classi di storage pre-CSI.
- Aggiornato il numero minimo di Kubernetes supportati a 1,22.

Modifiche nel 23,04

> Force volume Detach for ONTAP-SAN-* Volumes è supportato solo con le versioni di Kubernetes con la funzionalità non-Graceal Node Shutdown abilitata. La funzione Force Detach deve essere attivata al momento dell'installazione utilizzando `--enable-force-detach` Flag del programma di installazione Trident.

Correzioni

- Fixed Trident Operator to Use IPv6 localhost for installation when specified in spec.

- Sono stati corretti i permessi del ruolo del cluster Trident Operator per essere sincronizzati con i permessi del bundle ("Numero 799").
- Risolto il problema relativo al collegamento di un volume di blocco raw su più nodi in modalità RWX.
- Supporto corretto della clonazione FlexGroup e importazione di volumi per volumi SMB.
- Risolto il problema a causa del quale il controller Trident non poteva spegnersi immediatamente ("Numero 811").
- Aggiunta correzione per elencare tutti i nomi di igroup associati a un LUN specificato fornito con i driver ontap-san-*.
- Aggiunta di una correzione per consentire l'esecuzione di processi esterni fino al completamento.
- Corretto errore di compilazione per l'architettura s390 ("Numero 537").
- Corretto livello di registrazione errato durante le operazioni di montaggio del volume ("Numero 781").
- Risolto il potenziale errore di asserzione del tipo ("Numero 802").

Miglioramenti

- Kubernetes:
  - Aggiunto supporto per Kubernetes 1.27.
  - Aggiunto supporto per l'importazione di volumi LUKS.
  - Aggiunto supporto per la modalità di accesso al PVC ReadWriteOncePod.
  - Aggiunto il supporto per force Detach per volumi ONTAP-SAN-* durante scenari di non-Graged Node Shutdown.
  - Tutti i volumi ONTAP-SAN-* ora utilizzeranno igroups per nodo. Le LUN verranno mappate solo agli igroups mentre vengono pubblicate attivamente su tali nodi per migliorare la nostra posizione in materia di sicurezza. I volumi esistenti verranno opportunamente trasferiti al nuovo schema di igroup quando Trident stabilisce che è sicuro farlo senza influire sui carichi di lavoro attivi ("Numero 758").
  - Sicurezza Trident migliorata grazie alla pulizia degli igroups gestiti da Trident inutilizzati dai backend ONTAP-SAN-*.
- Aggiunto supporto per volumi SMB con Amazon FSX ai driver di storage ontap-nas-Economy e ontap-nas-Flexgroup.
- Supporto aggiunto per le condivisioni SMB con i driver di storage ontap-nas, ontap-nas-Economy e ontap-nas-Flexgroup.
- Aggiunto supporto per i nodi arm64 ("Numero 732").
- Miglioramento della procedura di shutdown di Trident disattivando prima i server API ("Numero 811").
- Aggiunto supporto di build multipiattaforma per host Windows e arm64 a Makefile; vedere BUILD.MD.

Dipendenze

Kubernetes: gli igroups con ambito backend non verranno più creati durante la configurazione dei driver ontap-san e ontap-san-Economy ("Numero 758").

Cambiamenti nel 23.01.1

Correzioni

- Fixed Trident Operator to Use IPv6 localhost for installation when specified in spec.

- Sono stati corretti i permessi del ruolo del cluster Trident Operator per essere sincronizzati con le autorizzazioni del bundle "Numero 799".
- Aggiunta di una correzione per consentire l'esecuzione di processi esterni fino al completamento.
- Risolto il problema relativo al collegamento di un volume di blocco raw su più nodi in modalità RWX.
- Supporto corretto della clonazione FlexGroup e importazione di volumi per volumi SMB.

Cambiamenti nel 23.01

Kubernetes 1.27 è ora supportato in Trident. Aggiornare Astra Trident prima di aggiornare Kubernetes.

Correzioni

- Kubernetes: Aggiunta di opzioni per escludere la creazione della policy di sicurezza Pod per correggere le installazioni Trident tramite Helm ("Numeri 783, 794").

Miglioramenti

Kubernetes

- Aggiunto supporto per Kubernetes 1.26.
- Migliore utilizzo delle risorse RBAC di Trident ("Numero 757").
- Aggiunta dell'automazione per rilevare e correggere sessioni iSCSI interrotte o obsolete sui nodi host.
- Aggiunto supporto per l'espansione dei volumi crittografati con LUKS.
- Kubernetes: Aggiunto il supporto della rotazione delle credenziali per i volumi crittografati LUKS.

Astra Trident

- Aggiunto supporto per volumi SMB con Amazon FSX per ONTAP al driver di storage ontap-nas.
- Aggiunto supporto per le autorizzazioni NTFS quando si utilizzano volumi SMB.
- Aggiunto supporto per pool di storage per volumi GCP con livello di servizio CVS.
- Aggiunto supporto per l'utilizzo opzionale di flexgroupAggregateList durante la creazione di FlexGroups con il driver di storage ontap-nas-flexgroup.
- Performance migliorate per il driver di storage ontap-nas-economy durante la gestione di più FlexVol.
- Aggiornamenti dataLIF abilitati per tutti i driver di storage NAS ONTAP.
- È stata aggiornata la convenzione di denominazione di Trident Deployment e DemonSet per riflettere il sistema operativo del nodo host.

Dipendenze

- Kubernetes: Aggiornato il numero minimo di Kubernetes supportati a 1.21.
- Non specificare più le LIF dei dati durante la configurazione `ontap-san` oppure `ontap-san-economy` driver.

Cambiamenti nel 22.10

Prima di eseguire l'aggiornamento ad Astra Trident 22.10, è necessario leggere le seguenti informazioni critiche.

** informazioni sulle di Astra Trident 22.10**

- Kubernetes 1.25 è ora supportato in Trident. Devi aggiornare Astra Trident alla versione 22.10 prima di eseguire l'aggiornamento a Kubernetes 1.25.
- Astra Trident applica ora rigorosamente l'utilizzo della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  Utilizzo di configurazioni o utilizzo non multipathing di `find_multipaths: yes` oppure `find_multipaths: smart` il valore nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'uso di `find_multipaths: no` dalla release 21.07.

Correzioni

- Risolto il problema specifico del backend ONTAP creato con `credentials` il campo non riesce a entrare in linea durante l'aggiornamento 22.07.0 ("Numero 759").
- **Docker:** risolto un problema che causava il mancato avvio del plug-in del volume Docker in alcuni ambienti ("Numero 548" e. "Numero 760").
- Risolto il problema SLM specifico dei backend SAN ONTAP per garantire la pubblicazione solo di un sottoinsieme di dati LIF appartenenti ai nodi di reporting.
- Risolto il problema delle performance in cui si verificavano scansioni non necessarie per LUN iSCSI durante il collegamento di un volume.
- Sono stati rimossi tentativi granulari all'interno del workflow iSCSI di Astra Trident per accelerare i guasti e ridurre gli intervalli di tentativi esterni.
- Risolto un problema a causa del quale si verificava un errore durante lo spurgo di un dispositivo iSCSI quando il dispositivo multipath corrispondente era già stato svuotato.

Miglioramenti

- Kubernetes:
  - Aggiunto supporto per Kubernetes 1.25. Devi aggiornare Astra Trident alla versione 22.10 prima di eseguire l'aggiornamento a Kubernetes 1.25.
  - Aggiunta di un ServiceAccount, ClusterRole e ClusterRoleBinding separato per la distribuzione Trident e DemonSet per consentire futuri miglioramenti delle autorizzazioni.
  - Supporto aggiunto per "condivisione di volumi tra spazi dei nomi".
- Tutti i Trident `ontap-*` I driver di storage ora funzionano con l'API REST di ONTAP.
- Aggiunto nuovo operatore yaml (`bundle_post_1_25.yaml`) senza un `PodSecurityPolicy` Per supportare Kubernetes 1.25.
- Aggiunto "Supporto per volumi con crittografia LUKS" per `ontap-san` e. `ontap-san-economy` driver di storage.
- Aggiunto supporto per nodi Windows Server 2019.
- Aggiunto "Supporto per volumi SMB su nodi Windows" tramite il `azure-netapp-files` driver di storage.
- Il rilevamento automatico dello switchover MetroCluster per i driver ONTAP è ora generalmente disponibile.

Dipendenze

- **Kubernetes:** aggiornato il numero minimo di Kubernetes supportati a 1.20.

- Driver ADS (Astra Data Store) rimosso.
- Supporto rimosso per `yes` e. `smart` opzioni per `find_multipaths` Durante la configurazione del multipathing del nodo di lavoro per iSCSI.

Cambiamenti nel 22.07

Correzioni

Kubernetes

- Risolto il problema della gestione dei valori booleani e numerici per il selettore di nodi durante la configurazione di Trident con Helm o l'operatore Trident. ("Numero GitHub 700")
- Risolto il problema di gestione degli errori dal percorso non CHAP, in modo che il kubelet ritenta in caso di errore. "Numero GitHub 736")

Miglioramenti

- Transizione da k8s.gcr.io a registry.k8s.io come registro predefinito per le immagini CSI
- I volumi ONTAP-SAN ora utilizzeranno igroups per nodo e mapperanno solo le LUN agli igroups mentre vengono attivamente pubblicate su tali nodi per migliorare la nostra posizione di sicurezza. I volumi esistenti verranno opportunamente trasferiti al nuovo schema di igroup quando Astra Trident stabilisce che è sicuro farlo senza influire sui carichi di lavoro attivi.
- Incluso un ResourceQuota con installazioni Trident per garantire che Trident DemonSet venga pianificato quando il consumo di PriorityClass è limitato per impostazione predefinita.
- Aggiunto il supporto per le funzioni di rete al driver Azure NetApp Files. ("Numero GitHub 717")
- Aggiunta dell'anteprima tecnica per il rilevamento automatico dello switchover MetroCluster ai driver ONTAP. ("Numero GitHub 228")

Dipendenze

- **Kubernetes:** aggiornato il numero minimo di Kubernetes supportati a 1.19.
- La configurazione back-end non consente più l'utilizzo di più tipi di autenticazione in una singola configurazione.

Rimozioni

- Il driver CVS AWS (obsoleto dal 22.04) è stato rimosso.
- Kubernetes
  - Rimozione della funzionalità SYS_ADMIN non necessaria dai pod di nodi.
  - Riduce il nodeprep fino a semplici informazioni host e rilevamento attivo del servizio per fare il meglio Conferma che i servizi NFS/iSCSI sono disponibili sui nodi di lavoro.

Documentazione

Un nuovo "Standard di sicurezza Pod" (PSS) è stata aggiunta la sezione che descrive i permessi abilitati da Astra Trident durante l'installazione.

Cambiamenti nel 22.04

NetApp continua a migliorare e migliorare i propri prodotti e servizi. Ecco alcune delle funzionalità più recenti di Astra Trident. Per le versioni precedenti, fare riferimento a. "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

- Analisi migliorata dei nomi degli iniziatori iSCSI. ("Numero GitHub 681")
- Risolto il problema a causa del quale i parametri della classe di storage CSI non erano consentiti. ("Numero GitHub 598")
- È stata corretta la dichiarazione della chiave duplicata in Trident CRD. ("Numero GitHub 671")
- Sono stati corretti registri Snapshot CSI imprecisi. ("Numero GitHub 629"))
- Risolto il problema di annullamento della pubblicazione dei volumi sui nodi cancellati. ("Numero GitHub 691")
- Aggiunta la gestione delle incoerenze del file system sui dispositivi a blocchi. ("Numero GitHub 656")
- Risolto il problema di recupero delle immagini con supporto automatico durante l'impostazione di `imageRegistry` flag durante l'installazione. ("Numero GitHub 715")
- Risolto il problema a causa del quale il driver Azure NetApp Files non riusciva a clonare un volume con più regole di esportazione.

Miglioramenti

- Le connessioni in entrata agli endpoint sicuri di Trident ora richiedono almeno TLS 1.3. ("Numero GitHub 698")
- Trident aggiunge ora gli header HSTS alle risposte dai suoi endpoint sicuri.
- Trident ora tenta di attivare automaticamente la funzione di permessi unix di Azure NetApp Files.
- **Kubernetes**: Trident demonset ora funziona con la classe di priorità system-node-critical. ("Numero GitHub 694")

Rimozioni

Il driver e-Series (disattivato dal 20.07) è stato rimosso.

Cambiamenti nel 22.01.1

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

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.
- Corretto il driver Azure NetApp Files per consentire nomi di risorse duplicati e limitare le operazioni a un'unica posizione.
- Corretto il driver Azure NetApp Files per consentire nomi di risorse duplicati e limitare le operazioni a un'unica 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 il supporto per l'opzione 'unixPermissions' ai volumi Azure NetApp Files. ("Numero GitHub 666")

Dipendenze

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

Cambiamenti nel 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

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.

- Quando si aggiorna un cluster Kubernetes dalla versione 1.24 alla 1.25 o successiva su cui è installato

Astra Trident, è necessario aggiornare values.yaml per impostarlo `excludePodSecurityPolicy` a. `true` oppure aggiungi `--set excludePodSecurityPolicy=true` al `helm upgrade` prima di aggiornare il cluster.

- 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]`.

  Impossibile specificare `dataLIF` Su un backend SAN ONTAP. Astra Trident rileva tutte le LIF iSCSI disponibili e le utilizza per stabilire la sessione multipath.

- Se si utilizza `solidfire-san` Driver con OpenShift 4.5, assicurarsi che i nodi di lavoro sottostanti utilizzino MD5 come algoritmo di autenticazione CHAP. Gli algoritmi CHAP conformi a FIPS sicuri SHA1, SHA-256 e SHA3-256 sono disponibili con Element 12.7.

Trova ulteriori informazioni

- "Astra Trident GitHub"

- "Blog di Astra Trident"

Versioni precedenti della documentazione

Se non utilizzi Astra Trident 24,02, la documentazione per le release precedenti è disponibile in base a. "Ciclo di vita del supporto di Astra Trident".

- "Astra Trident 23,10"
- "Astra Trident 23,07"
- "Astra Trident 23,04"
- "Astra Trident 23.01"
- "Astra Trident 22.10"
- "Astra Trident 22.07"
- "Astra Trident 22.04"

Inizia subito

Scopri Astra Trident

Scopri Astra Trident

Astra Trident è un progetto open source completamente supportato, gestito da NetApp nell'ambito di "Famiglia di prodotti Astra". È stato progettato per aiutare a soddisfare le richieste di persistenza delle applicazioni containerizzate utilizzando interfacce standard del settore, come Container Storage Interface (CSI).

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 esegue il provisioning e fornisce storage persistente dei container basato su Astra Trident. Offre inoltre funzionalità avanzate di gestione dei dati integrate con l'applicazione, come snapshot, backup e ripristino, log delle attività e cloning attivo per data Protection, disaster/recovery dei dati, audit dei dati e casi d'utilizzo della migrazione per i workload Kubernetes.

Scopri di più "Astra o registrati per una prova gratuita".

Che cos'è Astra Trident?

Astra Trident permette il consumo e la gestione delle risorse di storage in tutte le più apprezzate piattaforme di storage NetApp, nel cloud pubblico o on-premise, incluso ONTAP (AFF, FAS, Select, cloud, Amazon FSX per NetApp ONTAP), software Element (NetApp HCI, SolidFire), servizio Azure NetApp Files e Cloud Volumes Service su Google Cloud.

Astra Trident è un orchestrator di storage dinamico conforme a Container Storage Interface (CSI) che si integra in modo nativo con "Kubernetes". Astra Trident viene eseguito come singolo pod controller e un pod nodo su ogni nodo di lavoro nel cluster. Fare riferimento a. "Architettura Astra Trident" per ulteriori informazioni.

Astra Trident fornisce anche un'integrazione diretta con l'ecosistema Docker per le piattaforme di storage NetApp. Il plug-in volume Docker (nDVP) di NetApp supporta il provisioning e la gestione delle risorse storage dalla piattaforma storage agli host Docker. Fare riferimento a. "Implementa Astra Trident per Docker" per ulteriori informazioni.

Se è la prima volta che utilizzi Kubernetes, dovresti familiarizzare con il "Concetti e strumenti di Kubernetes".

Fai il test drive di Astra Trident

Per effettuare un test di guida, Richiedi l'accesso al menu "implementazione semplice e clonazione dello storage persistente per i workload containerizzati" "Test drive di NetApp" utilizzo di un'immagine di laboratorio pronta all'uso. Il test drive fornisce un ambiente sandbox con un cluster Kubernetes a tre nodi e Astra Trident installato e configurato. Si tratta di un ottimo modo per familiarizzare con Astra Trident ed esplorare le sue funzionalità.

Un'altra opzione è la "Guida all'installazione di kubeadm" Fornito da Kubernetes.

Non usare un cluster Kubernetes creato usando queste istruzioni in un ambiente di produzione. Utilizza le guide all'implementazione in produzione fornite dalla tua distribuzione per i cluster pronti per la produzione.

Integrazione di Kubernetes con prodotti NetApp

Il portfolio NetApp di prodotti storage si integra con molti aspetti di un cluster Kubernetes, fornendo funzioni avanzate di gestione dei dati, che migliorano funzionalità, capacità, performance e disponibilità dell'implementazione Kubernetes.

Amazon FSX per NetApp ONTAP

"Amazon FSX per NetApp ONTAP" È un servizio AWS completamente gestito che ti consente di lanciare ed eseguire file system basati sul sistema operativo per lo storage NetApp ONTAP.

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 ONTAP

"Cloud Volumes ONTAP" È un'appliance di storage solo software che esegue il software di gestione dei dati ONTAP nel cloud.

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.

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.

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.

Astra Trident può eseguire il provisioning e gestire i dispositivi di storage per applicazioni containerizzate direttamente sulla piattaforma di storage NetApp HCI sottostante.

NetApp ONTAP

"NetApp ONTAP" NetApp è un sistema operativo per lo storage unificato e multiprotocollo 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). Astra Trident supporta questi modelli di implementazione di ONTAP.

Per ulteriori informazioni

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

Architettura Astra Trident

Astra Trident viene eseguito come singolo pod controller e un pod nodo su ogni nodo di lavoro nel cluster. Il pod nodo deve essere in esecuzione su qualsiasi host in cui si desidera montare un volume Astra Trident.

Comprensione dei pod controller e dei pod di nodi

Astra Trident può essere implementato come un'unica soluzione Pod controller Trident e uno o più Pod di nodi Trident Sul cluster Kubernetes e utilizza Kubernetes *CSI Sidecar Containers* standard per semplificare l'implementazione dei plug-in CSI. "Kubernetes CSI Sidecar Containers" Sono mantenuti dalla community dello storage Kubernetes.

Kubernetes "selettori di nodi" e. "tollerazioni e contamini" vengono utilizzati per vincolare l'esecuzione di un pod su un nodo specifico o preferito. Puoi configurare selettori di nodo e tolleranze per controller e pod di nodo durante l'installazione di Astra Trident.

- Il plug-in del controller gestisce il provisioning e la gestione dei volumi, ad esempio snapshot e ridimensionamento.
- Il plug-in del nodo gestisce il collegamento dello storage al nodo.

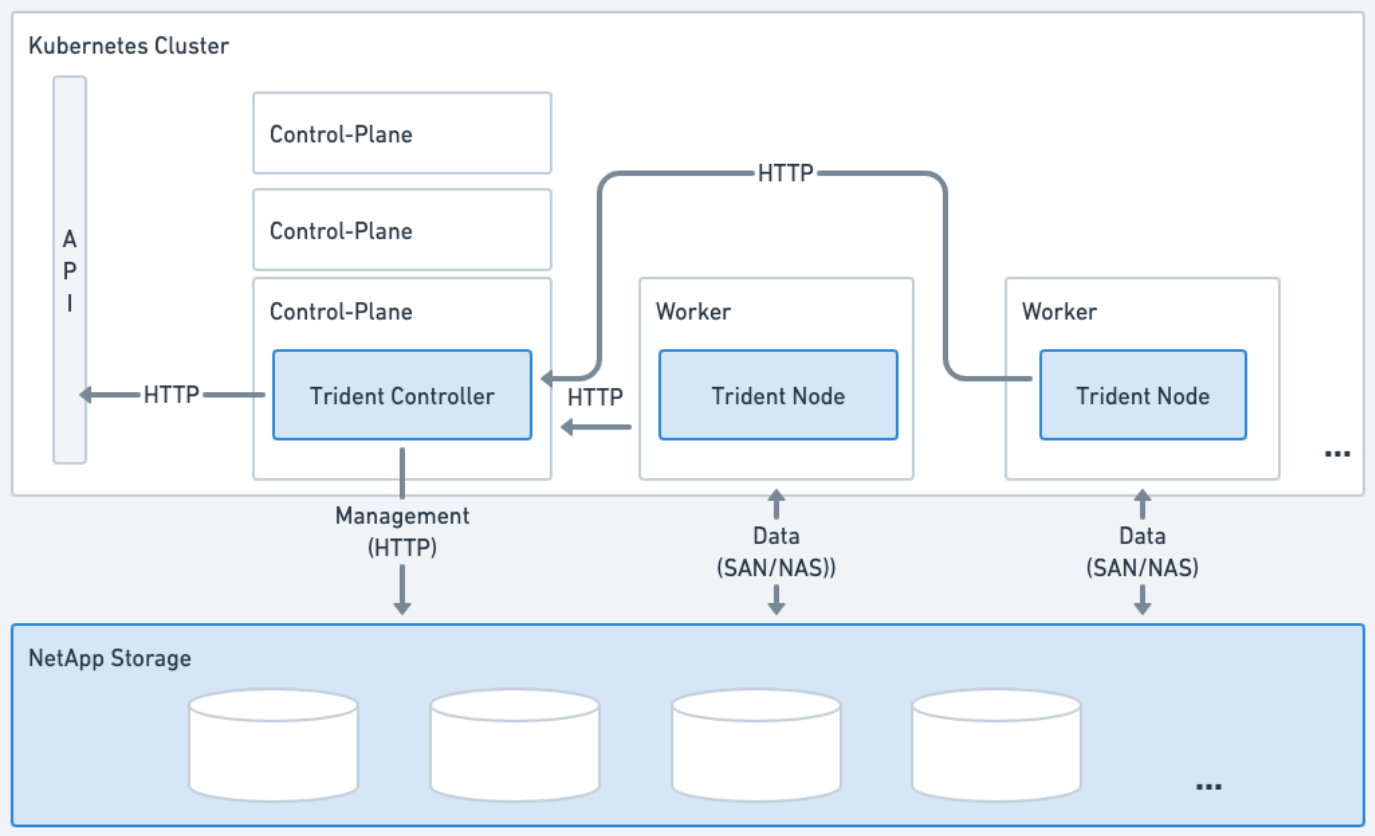


Figura 1. Astra Trident è implementato nel cluster Kubernetes

Pod controller Trident

Il controller Pod Trident è un singolo pod che esegue il plugin del controller CSI.

- Responsabile del provisioning e della gestione dei volumi nello storage NetApp
- Gestito da un'implementazione Kubernetes
- Può essere eseguito sul piano di controllo o sui nodi di lavoro, a seconda dei parametri di installazione.

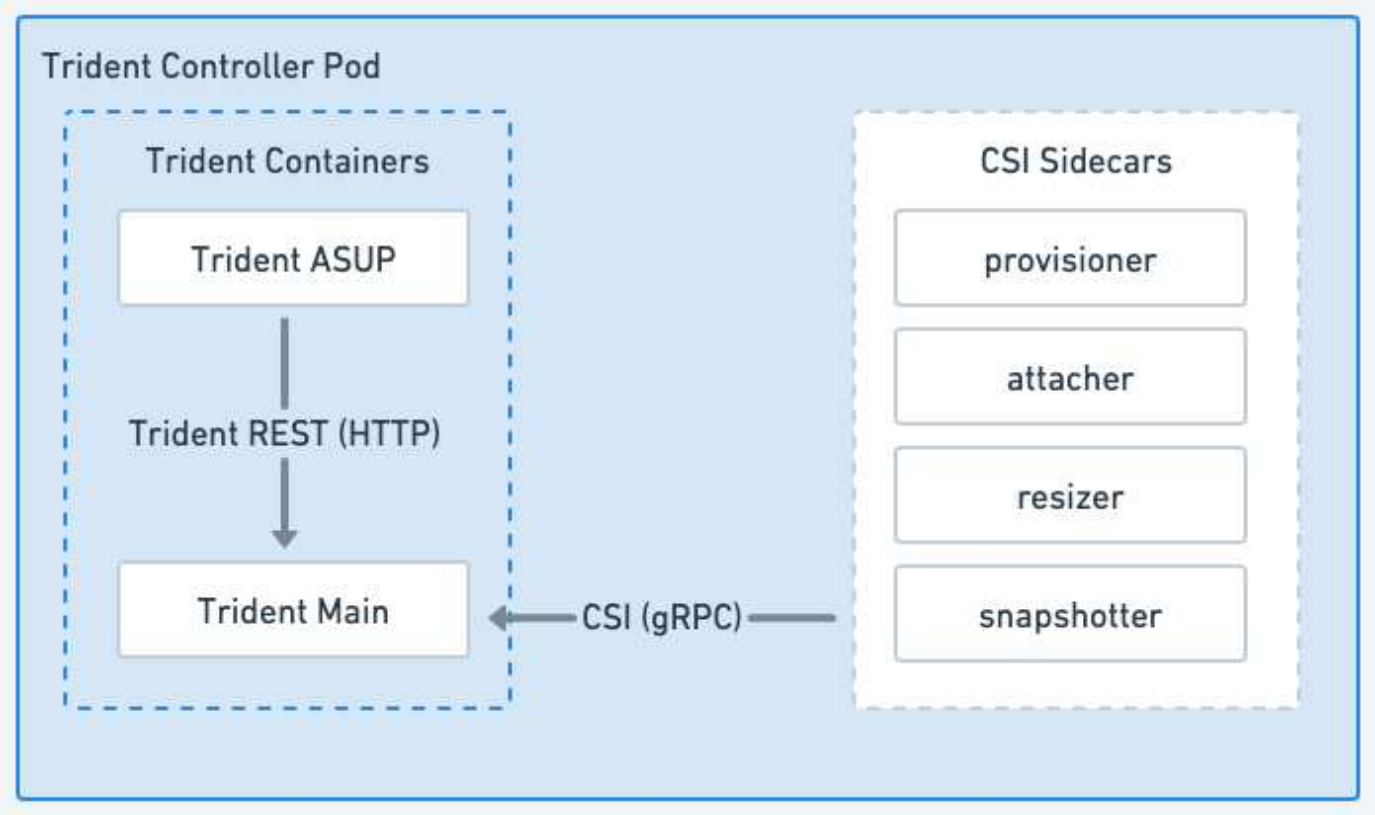


Figura 2. Diagramma del pod controller Trident

Pod di nodi Trident

I pod nodo Trident sono pod privilegiati che eseguono il plug-in nodo CSI.

- Responsabile del montaggio e dello smontaggio dello spazio di archiviazione per i pod in esecuzione sull'host
- Gestito da un Kubernetes DaemonSet
- Deve essere eseguito su qualsiasi nodo che monterà lo storage NetApp

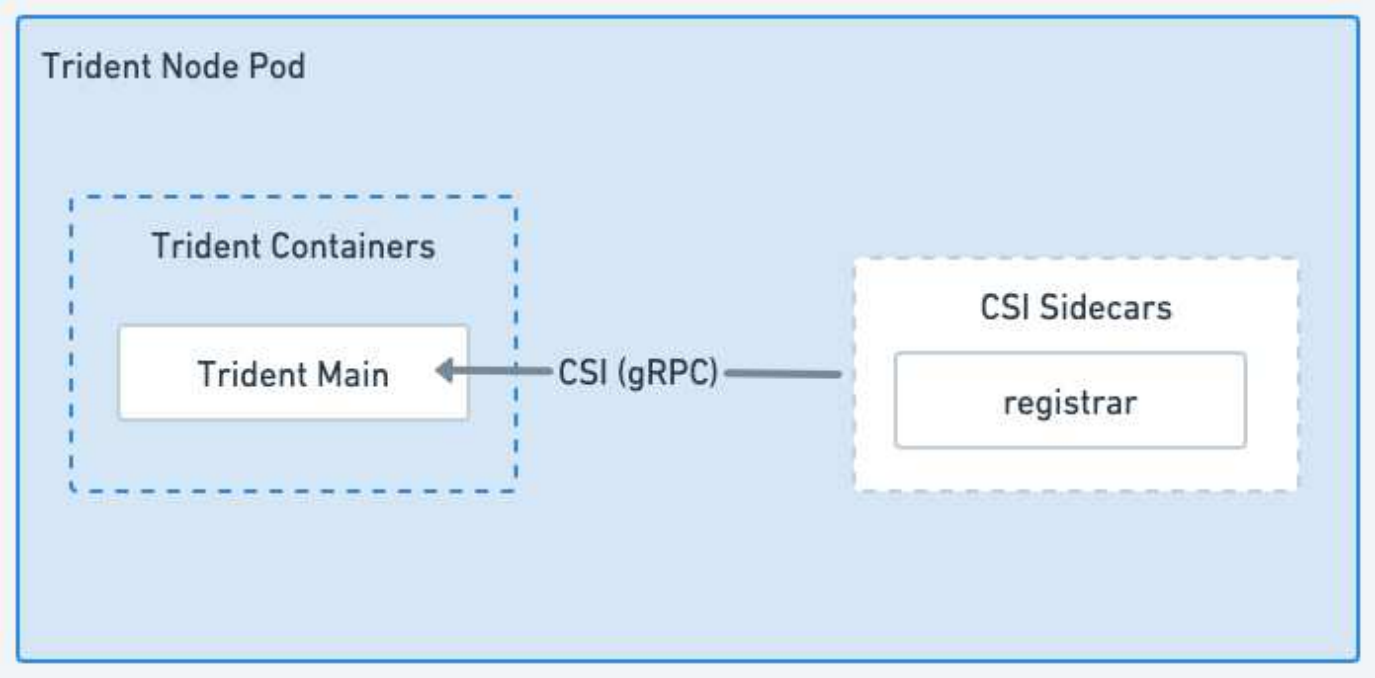


Figura 3. Diagramma del pod nodo Trident

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

Concetti

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 stessa di un volume e richiede la scelta di un pool di storage tra quelli associati alla classe di storage del volume in sospeso.

Associazione di classe storage

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.

Creazione di volumi

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.

Scopri di più sulla creazione di snapshot di volumi

- 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 NetApp Snapshot offre più stabilità, scalabilità, ripristinabilità e performance rispetto alle tecnologie Snapshot concorrenti. 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 NetApp Snapshot offre più stabilità, scalabilità, ripristinabilità e performance rispetto alle tecnologie Snapshot concorrenti. 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 ONTAP FlexClone consente di creare copie anche dei set di dati più estesi in maniera quasi istantanea. 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.

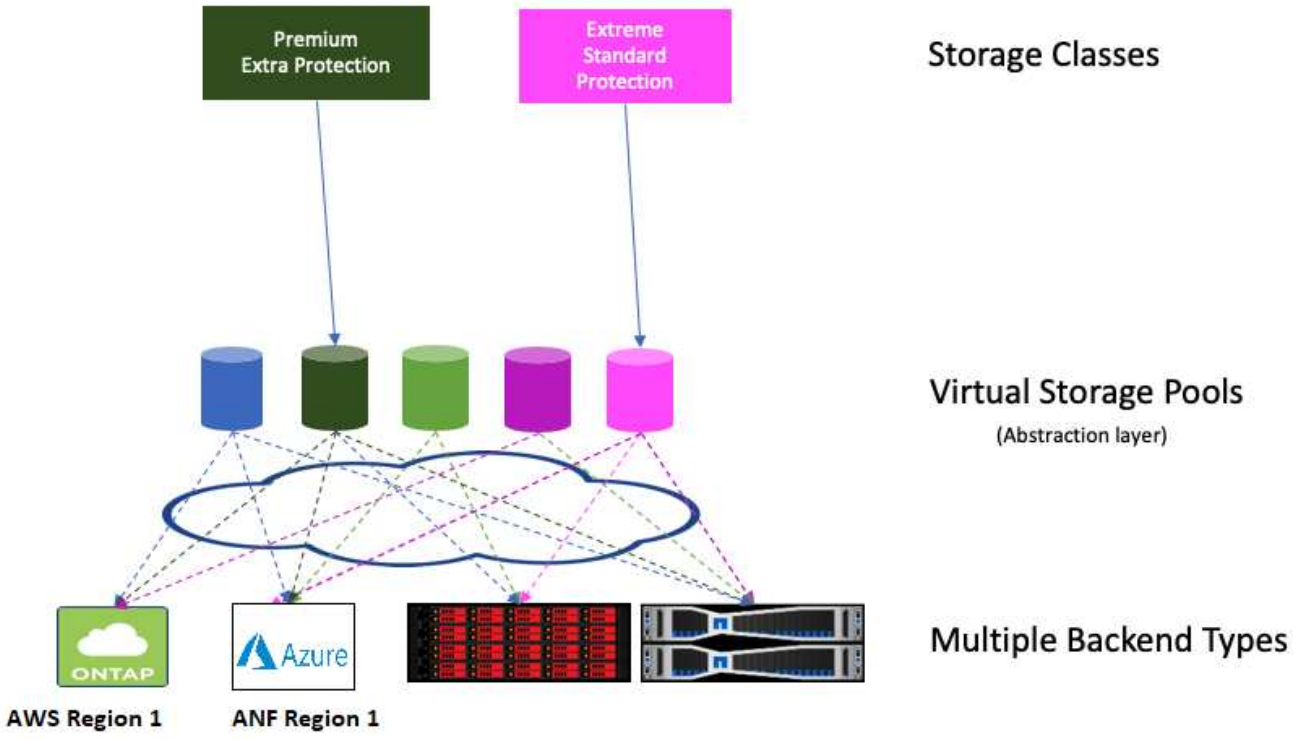
Con Astra Trident, puoi utilizzare VolumeSnapshots per creare nuovi PVS da essi. La creazione di PVS da queste snapshot viene eseguita utilizzando la tecnologia FlexClone per i backend ONTAP e CVS supportati. Quando si crea un PV da uno snapshot, il volume di backup è un FlexClone del volume principale 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 virtuali

I pool virtuali forniscono un livello di astrazione tra i backend di storage 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.

Informazioni sui pool virtuali

L'amministratore dello storage può definire pool 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 virtuali, non tentare di riorganizzare l'ordine dei pool virtuali esistenti in una definizione di back-end.
- Si consiglia di non modificare gli attributi per un pool virtuale esistente. È necessario definire un nuovo pool virtuale per apportare modifiche.

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. Per comodità, gli amministratori dello storage possono definire le etichette per ogni pool virtuale e raggruppare i volumi per etichetta.

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.

Informazioni sui gruppi di accesso ai volumi

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 gli stessi gruppi di accesso. I gruppi di accesso devono esistere prima che il backend dello storage venga aggiunto a Astra Trident e devono contenere gli IQN iSCSI da ogni nodo nel cluster Kubernetes che potrebbero potenzialmente montare i volumi con provisioning da quel 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 i limiti più recenti sui gruppi di accesso ai volumi, fare riferimento alla sezione "qui".

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

Avvio rapido di Astra Trident

Puoi installare Astra Trident e iniziare a gestire le risorse di storage in pochi passaggi. Prima di iniziare, rivedere "Requisiti di Astra Trident".

Per Docker, fare riferimento a. "Astra Trident per Docker".

Installare Astra Trident

Astra Trident offre diversi metodi di installazione e modalità ottimizzati per una vasta gamma di ambienti e organizzazioni.

"Installare Astra Trident"

Preparare il nodo di lavoro

Tutti i nodi di lavoro nel cluster Kubernetes devono essere in grado di montare i volumi forniti per i pod.

"Preparare il nodo di lavoro"

Creare un 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.

"Configurare un backend" per il tuo sistema storage

Creare una classe di storage Kubernetes

L'oggetto Kubernetes StorageClass specifica Astra Trident come provisioner e ti consente di creare una classe storage per eseguire il provisioning dei volumi con attributi personalizzabili. Astra Trident crea una classe di storage corrispondente per gli oggetti Kubernetes che specificano il provisioner Astra Trident.

"Creare una classe di storage"

Provisioning di un volume

Un *PersistentVolume* (PV) è una risorsa di storage fisico con provisioning eseguito dall'amministratore del cluster in un cluster Kubernetes. *PersistentVolumeClaim* (PVC) è una richiesta di accesso a PersistentVolume sul cluster.

Creare un PersistentVolume (PV) e un PersistentVolumeClaim (PVC) che utilizza Kubernetes StorageClass configurato per richiedere l'accesso al PV. È quindi possibile montare il PV su un pod.

"Provisioning di un volume"

Quali sono le prossime novità?

Da oggi puoi aggiungere backend aggiuntivi, gestire classi di storage, gestire i backend ed eseguire operazioni in termini di volume.

Requisiti

Prima di installare Astra Trident, è necessario esaminare questi requisiti generali di sistema. I backend specifici potrebbero avere requisiti aggiuntivi.

Informazioni critiche su Astra Trident

È necessario leggere le seguenti informazioni critiche su Astra Trident.

** informazioni su Astra **

- Kubernetes 1,29 è ora supportato in Trident. Aggiorna Astra Trident prima di eseguire l'upgrade di Kubernetes.
- Astra Trident impone rigorosamente l'utilizzo della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  Utilizzo di configurazioni o utilizzo non multipathing di `find_multipaths: yes` oppure `find_multipaths: smart` il valore nel file multipath.conf causerà errori di montaggio. Astra Trident ha consigliato di farlo `find_multipaths: no` dalla release 21.07.

Frontend supportati (orchestratori)

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

- Anthos on-premise (VMware) e anthos su Bare Metal 1,16
- Kubernetes 1,23 - 1,29
- OpenShift 4,10 - 4,15

L'operatore Trident è supportato con le seguenti versioni:

- Anthos on-premise (VMware) e anthos su Bare Metal 1,16
- Kubernetes 1,23 - 1,29
- OpenShift 4,10 - 4,15

Astra Trident funziona anche con un host di altre offerte Kubernetes completamente gestite e gestite in autonomia, inclusi Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service (AKS), Mirantis Kubernetes Engine (MKE), Rancher e VMware Tanzu Portfolio.

Astra Trident e ONTAP possono essere utilizzati come storage provider per "KubeVirt".

Prima di aggiornare un cluster Kubernetes da 1,24 a 1,25 o versioni successive con Astra Trident installato, fare riferimento a. "Aggiornare un'installazione Helm".

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
- Cloud Volumes ONTAP
- Cloud Volumes Service per GCP
- FAS/AFF/Select 9.5 o versione successiva
- Array All SAN (ASA) NetApp
- Software NetApp HCI/Element 11 o superiore

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?
Astra Trident	1,23 - 1,29	No
Snapshot dei volumi	1,23 - 1,29	No
PVC dalle istantanee dei volumi	1,23 - 1,29	No
Ridimensionamento di iSCSI PV	1,23 - 1,29	No
CHAP bidirezionale ONTAP	1,23 - 1,29	No
Policy di esportazione dinamiche	1,23 - 1,29	No
Operatore Trident	1,23 - 1,29	No
Topologia CSI	1,23 - 1,29	No

Sistemi operativi host testati

Sebbene Astra Trident non supporti ufficialmente sistemi operativi specifici, sono noti i seguenti elementi:

- Versioni di RedHat CoreOS (RHCOS) supportate da OpenShift Container Platform (AMD64 e ARM64)
- RHEL 8+ (AMD64 E ARM64)

NVMe/TCP richiede RHEL 9 o versione successiva.

- Ubuntu 22.04 o versione successiva (AMD64 e ARM64)
- Windows Server 2019 (AMD64)

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.

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

Configurazione dell'host

Tutti i nodi di lavoro nel cluster Kubernetes devono essere in grado di montare i volumi forniti per i pod. Per preparare i nodi di lavoro, devi installare i tool NFS, iSCSI o NVMe in base alla tua selezione di driver.

"Preparare il nodo di lavoro"

Configurazione del sistema storage

Astra Trident potrebbe richiedere modifiche a un sistema storage prima che possa essere utilizzato da una configurazione di back-end.

"Configurare i backend"

Porte Astra Trident

Astra Trident richiede l'accesso a porte specifiche per la comunicazione.

"Porte Astra Trident"

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
v1.23.0	• docker.io/netapp/tridente:24.02.0 • docker.io/netapp/trident-autosupport:24,02 • registry.k8s.io/sig-storage/csi-provisioner:v4,0.0 • registry.k8s.io/sig-storage/csi-attacher:v4,5.0 • registry.k8s.io/sig-storage/csi-resizer:v1,9.3 • registry.k8s.io/sig-storage/csi-snapshotter:v6,3.3 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.10.0 • docker.io/netapp/trident-operator:24.02.0 (opzionale)
v1.24.0	• docker.io/netapp/tridente:24.02.0 • docker.io/netapp/trident-autosupport:24,02 • registry.k8s.io/sig-storage/csi-provisioner:v4,0.0 • registry.k8s.io/sig-storage/csi-attacher:v4,5.0 • registry.k8s.io/sig-storage/csi-resizer:v1,9.3 • registry.k8s.io/sig-storage/csi-snapshotter:v6,3.3 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.10.0 • docker.io/netapp/trident-operator:24.02.0 (opzionale)
v1.25.0	• docker.io/netapp/tridente:24.02.0 • docker.io/netapp/trident-autosupport:24,02 • registry.k8s.io/sig-storage/csi-provisioner:v4,0.0 • registry.k8s.io/sig-storage/csi-attacher:v4,5.0 • registry.k8s.io/sig-storage/csi-resizer:v1,9.3 • registry.k8s.io/sig-storage/csi-snapshotter:v6,3.3 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.10.0 • docker.io/netapp/trident-operator:24.02.0 (opzionale)

Versione di Kubernetes	Immagine container
v1.26.0	• docker.io/netapp/tridente:24.02.0 • docker.io/netapp/trident-autosupport:24,02 • registry.k8s.io/sig-storage/csi-provisioner:v4,0.0 • registry.k8s.io/sig-storage/csi-attacher:v4,5.0 • registry.k8s.io/sig-storage/csi-resizer:v1,9.3 • registry.k8s.io/sig-storage/csi-snapshotter:v6,3.3 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.10.0 • docker.io/netapp/trident-operator:24.02.0 (opzionale)
v1.27.0	• docker.io/netapp/tridente:24.02.0 • docker.io/netapp/trident-autosupport:24,02 • registry.k8s.io/sig-storage/csi-provisioner:v4,0.0 • registry.k8s.io/sig-storage/csi-attacher:v4,5.0 • registry.k8s.io/sig-storage/csi-resizer:v1,9.3 • registry.k8s.io/sig-storage/csi-snapshotter:v6,3.3 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.10.0 • docker.io/netapp/trident-operator:24.02.0 (opzionale)
v1.28.0	• docker.io/netapp/tridente:24.02.0 • docker.io/netapp/trident-autosupport:24,02 • registry.k8s.io/sig-storage/csi-provisioner:v4,0.0 • registry.k8s.io/sig-storage/csi-attacher:v4,5.0 • registry.k8s.io/sig-storage/csi-resizer:v1,9.3 • registry.k8s.io/sig-storage/csi-snapshotter:v6,3.3 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.10.0 • docker.io/netapp/trident-operator:24.02.0 (opzionale)

Versione di Kubernetes	Immagine container
v1.29.0	• docker.io/netapp/tridente:24.02.0 • docker.io/netapp/trident-autosupport:24,02 • registry.k8s.io/sig-storage/csi-provisioner:v4,0.0 • registry.k8s.io/sig-storage/csi-attacher:v4,5.0 • registry.k8s.io/sig-storage/csi-resizer:v1,9.3 • registry.k8s.io/sig-storage/csi-snapshotter:v6,3.3 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.10.0 • docker.io/netapp/trident-operator:24.02.0 (opzionale)

Installare Astra Trident

Installare utilizzando l’operatore Trident

Installare usando tridentctl

Utilizzare Astra Trident

Preparare il nodo di lavoro

Tutti i nodi di lavoro nel cluster Kubernetes devono essere in grado di montare i volumi forniti per i pod. Per preparare i nodi di lavoro, è necessario installare gli strumenti NFS, iSCSI o NVMe/TCP in base alla selezione del driver.

Selezionare gli strumenti giusti

Se si utilizza una combinazione di driver, è necessario installare tutti gli strumenti necessari per i driver. Le versioni recenti di RedHat CoreOS hanno gli strumenti installati di default.

Strumenti NFS

"Installare gli strumenti NFS" se si utilizza: `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `azure-netapp-files`, `gcp-cvs`.

Strumenti iSCSI

"Installare gli strumenti iSCSI" se si utilizza: `ontap-san`, `ontap-san-economy`, `solidfire-san`.

Strumenti NVMe

"Installazione degli strumenti NVMe" se si utilizza `ontap-san` Per il protocollo NVMe (nonvolatile Memory Express) su TCP (NVMe/TCP).

> Consigliamo ONTAP 9,12 o versione successiva per NVMe/TCP.

Rilevamento del servizio del nodo

Astra Trident tenta di rilevare automaticamente se il nodo può eseguire servizi iSCSI o NFS.

> Il rilevamento del servizio nodo identifica i servizi rilevati ma non garantisce che i servizi siano configurati correttamente. Al contrario, l'assenza di un servizio rilevato non garantisce il mancato funzionamento del montaggio del volume.

Rivedere gli eventi

Astra Trident crea eventi per il nodo per identificare i servizi rilevati. Per rivedere questi eventi, eseguire:

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

Esaminare i servizi rilevati

Astra Trident identifica i servizi abilitati per ciascun nodo sul nodo Trident CR. Per visualizzare i servizi rilevati, eseguire:

```
tridentctl get node -o wide -n <Trident namespace>
```

Volumi NFS

Installa gli strumenti NFS utilizzando i comandi del tuo sistema operativo. Assicurarsi che il servizio NFS venga avviato durante l'avvio.

RHEL 8+

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```

Riavviare i nodi di lavoro dopo aver installato gli strumenti NFS per evitare errori durante il collegamento dei volumi ai container.

Volumi iSCSI

Astra Trident può stabilire automaticamente una sessione iSCSI, eseguire la scansione delle LUN e rilevare i dispositivi multipath, formattarli e montarli su un pod.

Funzionalità di riparazione automatica di iSCSI

Per i sistemi ONTAP, Astra Trident esegue la riparazione automatica di iSCSI ogni cinque minuti per:

1. **Identificare** lo stato della sessione iSCSI desiderato e lo stato della sessione iSCSI corrente.
2. **Confrontare** lo stato desiderato con quello corrente per identificare le riparazioni necessarie. Astra Trident determina le priorità di riparazione e quando anticipare le riparazioni.
3. **Eseguire le riparazioni** necessarie per riportare lo stato della sessione iSCSI corrente allo stato della sessione iSCSI desiderato.

I registri delle attività di riparazione automatica si trovano in `trident-main` Container sul rispettivo pod Demonset. Per visualizzare i registri, è necessario aver impostato `debug` "True" durante l'installazione di Astra Trident.

Le funzionalità di riparazione automatica iSCSI di Astra Trident possono contribuire a prevenire:

- Sessioni iSCSI obsolete o non funzionanti che potrebbero verificarsi dopo un problema di connettività di rete. In caso di sessione obsoleta, Astra Trident attende sette minuti prima di disconnettersi per ristabilire la connessione con un portale.

  Ad esempio, se i segreti CHAP sono stati ruotati sul controller di storage e la rete perde la connettività, i vecchi segreti CHAP (*stale*) potrebbero persistere. L'autoriparazione è in grado di riconoscerlo e ristabilire automaticamente la sessione per applicare i segreti CHAP aggiornati.

- Sessioni iSCSI mancanti

- LUN mancanti

Installare gli strumenti iSCSI

Installare gli strumenti iSCSI utilizzando i comandi del sistema operativo.

Prima di iniziare

- Ogni nodo del cluster Kubernetes deve avere un IQN univoco. **Questo è un prerequisito necessario**.
- Se si utilizza RHCOS versione 4.5 o successiva, o un'altra distribuzione Linux compatibile con RHEL, con `solidfire-san` Driver ed Element OS 12.5 o versioni precedenti, assicurarsi che l'algoritmo di autenticazione CHAP sia impostato su MD5 in `/etc/iscsi/iscsid.conf`. Gli algoritmi CHAP conformi a FIPS sicuri SHA1, SHA-256 e SHA3-256 sono disponibili con Element 12.7.

```
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, specificare `discard` MountOption in StorageClass per eseguire la rigenerazione dello spazio inline. Fare riferimento a. "Documentazione RedHat".

RHEL 8+

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
```

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
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

> 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.

Configurare o disattivare la riparazione automatica iSCSI

Puoi configurare le seguenti impostazioni di riparazione automatica di iSCSI Astra Trident per correggere le sessioni obsolete:

- **Intervallo di autoriparazione iSCSI**: Determina la frequenza con cui viene richiamata l'autoriparazione iSCSI (valore predefinito: 5 minuti). È possibile configurare l'esecuzione più frequente impostando un numero minore o meno frequente impostando un numero maggiore.

Impostando l'intervallo di riparazione automatica iSCSI su 0 si arresta completamente la riparazione automatica iSCSI. Si sconsiglia di disattivare la funzionalità di riparazione automatica iSCSI; questa opzione deve essere disattivata solo in alcuni scenari quando la riparazione automatica iSCSI non funziona come previsto o a scopo di debug.

- **Tempo di attesa per la riparazione automatica iSCSI**: Determina la durata di attesa per la riparazione automatica iSCSI prima di uscire da una sessione non corretta e di tentare nuovamente l'accesso (valore predefinito: 7 minuti). È possibile configurarlo su un numero maggiore in modo che le sessioni identificate come non integre debbano attendere più a lungo prima di essere disconnesse e quindi venga effettuato un tentativo di riconnessione o un numero minore per disconnettersi e accedere in precedenza.

Timone

Per configurare o modificare le impostazioni di riparazione automatica iSCSI, passare il `iscsiSelfHealingInterval` e. `iscsiSelfHealingWaitTime` parametri durante l'installazione del timone o l'aggiornamento del timone.

Il seguente esempio imposta l'intervallo di riparazione automatica iSCSI su 3 minuti e il tempo di attesa di riparazione automatica su 6 minuti:

```
helm install trident trident-operator-100.2402.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

tridentctl

Per configurare o modificare le impostazioni di riparazione automatica iSCSI, passare il `iscsi-self-healing-interval` e. `iscsi-self-healing-wait-time` parametri durante l'installazione o l'aggiornamento di tridentctl.

Il seguente esempio imposta l'intervallo di riparazione automatica iSCSI su 3 minuti e il tempo di attesa di riparazione automatica su 6 minuti:

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

Volumi NVMe/TCP

Installa gli strumenti NVMe utilizzando i comandi del tuo sistema operativo.

- NVMe richiede RHEL 9 o versione successiva.
- Se la versione del kernel del nodo Kubernetes è troppo vecchia o se il pacchetto NVMe non è disponibile per la versione del kernel in uso, potrebbe essere necessario aggiornare la versione del kernel del nodo a una versione con il pacchetto NVMe.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Verificare l'installazione

Dopo l'installazione, verificare che ogni nodo nel cluster Kubernetes disponga di un NQN univoco utilizzando il comando:

```
cat /etc/nvme/hostnqn
```

Astra Trident modifica l' `ctrl_device_tmo` Value per garantire che NVMe non si arrenda sul percorso in caso di arresti. Non modificare questa impostazione.

Configurare e gestire i backend

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 offre automaticamente pool di storage da backend che soddisfano i requisiti definiti da una classe di storage. Scopri come configurare il back-end per il tuo sistema storage.

- "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"

Azure NetApp Files

Configurare un backend Azure NetApp Files

Puoi configurare Azure NetApp Files come back-end per Astra Trident. È possibile collegare volumi NFS e SMB utilizzando un backend Azure NetApp Files. Astra Trident supporta anche la gestione delle credenziali utilizzando identità gestite per i cluster Azure Kubernetes Services (AKS).

Dettagli del driver Azure NetApp Files

Astra Trident offre i seguenti driver di storage Azure NetApp Files per comunicare con il cluster. Le modalità di accesso supportate sono: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
`azure-netapp-files`	NFS PMI	Filesystem	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Considerazioni

- Il servizio Azure NetApp Files non supporta volumi inferiori a 100 GB. Astra Trident crea automaticamente volumi 100-GiB se viene richiesto un volume più piccolo.
- Astra Trident supporta volumi SMB montati su pod eseguiti solo su nodi Windows.

Identità gestite per AKS

Astra Trident supporta "identità gestite" Per i cluster di Azure Kubernetes Services. Per sfruttare al meglio la gestione semplificata delle credenziali offerta dalle identità gestite, è necessario disporre di:

- Un cluster Kubernetes implementato utilizzando AKS
- Identità gestite configurate sul cluster AKS kuBoost
- Astra Trident ha installato che include `cloudProvider` da specificare `"Azure"`.

Operatore Trident

Per installare Astra Trident usando l'operatore Trident, modifica `tridentorchestrator_cr.yaml` da impostare `cloudProvider` a. `"Azure"`. Ad esempio:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

Timone

Nell'esempio seguente vengono installati i set Astra Trident `cloudProvider` In Azure utilizzando la variabile di ambiente `$CP`:

```
helm install trident trident-operator-100.2402.0.tgz --create
-namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

<code>tridentctl</code>

Nell'esempio seguente viene installato Astra Trident e impostato l' `cloudProvider` contrassegna come `Azure`:

```
tridentctl install --cloud-provider="Azure" -n trident
```

Identità cloud per AKS

L'identità del cloud consente ai pod Kubernetes di accedere alle risorse Azure autenticandosi come identità del carico di lavoro invece di fornire credenziali Azure esplicite.

Per sfruttare l'identità cloud in Azure è necessario disporre di:

- Un cluster Kubernetes implementato utilizzando AKS
- Identità del workload e issuer oidc configurati nel cluster AKS Kubernetes
- Astra Trident ha installato che include `cloudProvider` da specificare `"Azure"` e. `cloudIdentity` specifica dell'identità del workload

Operatore Trident

Per installare Astra Trident usando l'operatore Trident, modifica `tridentorchestrator_cr.yaml` da impostare `cloudProvider` a. `"Azure"` e impostare `cloudIdentity` a. `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

Ad esempio:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  *cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx'*
```

Timone

Impostare i valori per i flag **cloud-provider (CP)** e **cloud-Identity (ci)** utilizzando le seguenti variabili di ambiente:

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx"
```

Nell'esempio seguente viene installato e impostato Astra Trident `cloudProvider` In Azure utilizzando la variabile di ambiente `$CP` e imposta `cloudIdentity` utilizzo della variabile di ambiente `$CI`:

```
helm install trident trident-operator-100.2402.0.tgz --set
cloudProvider=$CP --set cloudIdentity=$CI
```

<code>tridentctl</code>

Impostare i valori per i flag **cloud provider** e **cloud Identity** utilizzando le seguenti variabili di ambiente:

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx"
```

Nell'esempio seguente viene installato Astra Trident e impostato l' `cloud-provider` contrassegna come `$CP`, e. `cloud-identity` a. `$CI`:

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

Prepararsi a configurare un backend Azure NetApp Files

Prima di poter configurare il backend Azure NetApp Files, è necessario assicurarsi che siano soddisfatti i seguenti requisiti.

Prerequisiti per volumi NFS e SMB

Se si utilizza Azure NetApp Files per la prima volta o in una nuova posizione, è necessaria una configurazione iniziale per configurare Azure NetApp Files e creare un volume NFS. Fare riferimento a. "Azure: Configura Azure NetApp Files e crea un volume NFS".

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

- `subscriptionID`, `tenantID`, `clientID`, `location`, e. `clientSecret` Sono opzionali quando si utilizzano identità gestite su un cluster AKS.
- `tenantID`, `clientID`, e. `clientSecret` Sono opzionali quando si utilizza un'identità cloud su un cluster AKS.

- Un pool di capacità. Fare riferimento a. "Microsoft: Creare un pool di capacità per Azure NetApp Files".
- Una subnet delegata a Azure NetApp Files. Fare riferimento a. "Microsoft: Delegare una subnet a Azure NetApp Files".
- `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:
  - Il ruolo di Proprietario o collaboratore "Predefinito da Azure".
  - R "Ruolo di collaboratore personalizzato" a livello di abbonamento (`assignableScopes`) Con le seguenti autorizzazioni limitate solo a quanto richiesto da Astra Trident. Dopo aver creato il ruolo personalizzato, "Assegnare il ruolo utilizzando il portale Azure".

Ruolo collaboratore personalizzato

```
{
    "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
    "properties": {
        "roleName": "custom-role-with-limited-perms",
        "description": "custom role providing limited
permissions",
        "assignableScopes": [
            "/subscriptions/<subscription-id>"
        ],
        "permissions": [
            {
                "actions": [

"Microsoft.NetApp/netAppAccounts/capacityPools/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
                    "Microsoft.Network/virtualNetworks/read",

"Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
```

```
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/delete",
                    "Microsoft.Features/features/read",
                    "Microsoft.Features/operations/read",
                    "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
                ],
                "notActions": [],
                "dataActions": [],
                "notDataActions": []
            }
        ]
    }
}
```

- 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.
- Da utilizzare `Cloud Identity`, ottenere il `client ID` da un "identità gestita assegnata dall'utente" E specificare tale ID in `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

Requisiti aggiuntivi per i volumi SMB

Per creare un volume SMB, è necessario disporre di:

- Active Directory configurato e connesso a Azure NetApp Files. Fare riferimento a. "Microsoft: Creazione e gestione delle connessioni Active Directory per Azure NetApp Files".
- Un cluster Kubernetes con un nodo controller Linux e almeno un nodo di lavoro Windows che esegue Windows Server 2019. Astra Trident supporta volumi SMB montati su pod eseguiti solo su nodi Windows.
- Almeno un segreto di Astra Trident contenente le credenziali di Active Directory in modo che Azure NetApp Files possa autenticarsi in Active Directory. Per generare un segreto `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Proxy CSI configurato come servizio Windows. Per configurare un `csi-proxy`, fare riferimento a. "GitHub: Proxy CSI" oppure "GitHub: Proxy CSI per Windows" Per i nodi Kubernetes in esecuzione su Windows.

Opzioni di configurazione back-end Azure NetApp Files ed esempi

Scopri le opzioni di configurazione di back-end NFS e SMB per Azure NetApp Files e consulta gli esempi di configurazione.

Opzioni di configurazione back-end

Astra Trident utilizza la configurazione backend (subnet, rete virtuale, livello di servizio e posizione) per creare volumi Azure NetApp Files su pool di capacità disponibili nel percorso richiesto e corrispondere al livello di servizio e alla subnet richiesti.

Astra Trident non supporta i pool di capacità QoS manuali.

I backend Azure NetApp Files forniscono queste opzioni di configurazione.

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 Opzionale quando le identità gestite sono abilitate su un cluster AKS.	
`tenantID`	L'ID tenant di una registrazione app Opzionale quando si utilizzano identità gestite o identità cloud su un cluster AKS.	
`clientID`	L'ID client di una registrazione dell'applicazione Opzionale quando si utilizzano identità gestite o identità cloud su un cluster AKS.	
`clientSecret`	Il segreto del client da una registrazione dell'applicazione Opzionale quando si utilizzano identità gestite o identità cloud su un cluster AKS.	
`serviceLevel`	Uno di `Standard`, `Premium`, o. `Ultra`	"" (casuale)

Parametro	Descrizione	Predefinito
`location`	Nome della posizione di Azure in cui verranno creati i nuovi volumi Opzionale quando le identità gestite sono abilitate su un cluster AKS.	
`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`	""
`networkFeatures`	Serie di funzionalità VNET per un volume, potrebbe essere `Basic` oppure `Standard`. Le funzioni di rete non sono disponibili in tutte le regioni e potrebbero essere abilitate in un abbonamento. Specificare `networkFeatures` se la funzionalità non è attivata, il provisioning del volume non viene eseguito correttamente.	""
`nfsMountOptions`	Controllo dettagliato delle opzioni di montaggio NFS. Ignorato per i volumi SMB. Per montare i volumi utilizzando NFS versione 4.1, include `nfsvers=4` Nell'elenco delle opzioni di montaggio delimitate da virgole, scegliere NFS v4.1. Le opzioni di montaggio impostate in una definizione di classe di storage sovrascrivono le opzioni di montaggio impostate nella configurazione backend.	"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
`nasType`	Configurare la creazione di volumi NFS o SMB. Le opzioni sono `nfs`, `smb` o nullo. L'impostazione su Null consente di impostare i volumi NFS come predefiniti.	`nfs`

Per ulteriori informazioni sulle funzioni di rete, fare riferimento a. "Configurare le funzionalità di rete per un volume Azure NetApp Files".

Autorizzazioni e risorse richieste

Se viene visualizzato l'errore "Nessun pool di capacità trovato" durante la creazione di un PVC, è probabile che la registrazione dell'applicazione non disponga delle autorizzazioni e delle risorse necessarie (subnet, rete virtuale, pool di capacità) associate. Se il debug è attivato, Astra Trident registra le risorse Azure rilevate al momento della creazione del backend. Verificare che venga utilizzato un ruolo appropriato.

I valori per `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork`, e. `subnet` può essere specificato utilizzando nomi brevi o completi. Nella maggior parte dei casi, si consiglia di utilizzare nomi completi, in quanto i nomi brevi possono corrispondere a più risorse con lo stesso nome.

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 seguono questo 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>

Provisioning di volumi

È possibile controllare il provisioning del volume predefinito specificando le seguenti opzioni in una sezione speciale del file di configurazione. Fare riferimento a. Configurazioni di esempio per ulteriori informazioni.

Parametro	Descrizione	Predefinito
`exportRule`	Regole di esportazione per nuovi volumi. `exportRule` Deve essere un elenco separato da virgole di qualsiasi combinazione di indirizzi IPv4 o subnet IPv4 nella notazione CIDR. Ignorato per i volumi SMB.	"0.0.0.0/0"
`snapshotDir`	Controlla la visibilità della directory .snapshot	"falso"
`size`	La dimensione predefinita dei nuovi volumi	"100 G"
`unixPermissions`	Le autorizzazioni unix dei nuovi volumi (4 cifre ottali). Ignorato per i volumi SMB.	"" (funzione di anteprima, richiede la whitelist nell'abbonamento)

Configurazioni di esempio

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.

Configurazione minima

Questa è la configurazione backend minima assoluta. Con questa configurazione, Astra Trident scopre tutti gli account NetApp, i pool di capacità e le subnet delegate a Azure NetApp Files nel percorso configurato, e posiziona nuovi volumi in uno di tali pool e subnet in modo casuale. Perché `nasType` viene omesso, il `nfs` Viene applicato il valore predefinito e il backend eseguirà il provisioning dei volumi NFS.

Questa configurazione è l'ideale se stai iniziando a utilizzare Azure NetApp Files e provando qualcosa, ma in pratica vorresti fornire un ulteriore ambito per i volumi da te forniti.

```
---
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
```

Identità gestite per AKS

Questa configurazione di backend omette `subscriptionID`, `tenantID`, `clientID`, e. `clientSecret`, che sono opzionali quando si utilizzano identità gestite.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools: ["ultra-pool"]
  resourceGroups: ["aks-ami-eastus-rg"]
  netappAccounts: ["smb-na"]
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
```

Identità cloud per AKS

Questa configurazione di backend omette `tenantID`, `clientID`, e. `clientSecret`, che sono opzionali quando si utilizza un'identità cloud.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools: ["ultra-pool"]
  resourceGroups: ["aks-ami-eastus-rg"]
  netappAccounts: ["smb-na"]
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

Configurazione specifica del livello di servizio con filtri 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 scopre automaticamente tutte le subnet delegate a Azure NetApp Files in tale posizione e posiziona 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
```

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
networkFeatures: Standard
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'
```

Configurazione dei pool virtuali

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. Le etichette dei pool virtuali sono state utilizzate per differenziare i pool in base a. `performance`.

```
---
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
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
- labels:
    performance: gold
  serviceLevel: Ultra
  capacityPools:
  - ultra-1
  - ultra-2
  networkFeatures: Standard
- labels:
    performance: silver
  serviceLevel: Premium
  capacityPools:
  - premium-1
- labels:
    performance: bronze
  serviceLevel: Standard
  capacityPools:
  - standard-1
  - standard-2
```

Definizioni delle classi di storage

Quanto segue `StorageClass` le definizioni si riferiscono ai pool di storage sopra indicati.

Definizioni di esempio con `parameter.selector` campo

Utilizzo di `parameter.selector` è possibile specificare per ciascuno `StorageClass` il pool virtuale 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
```

Definizioni di esempio per volumi SMB

Utilizzo di `nasType`, `node-stage-secret-name`, e. `node-stage-secret-namespace`, È possibile specificare un volume SMB e fornire le credenziali Active Directory richieste.

Configurazione di base sullo spazio dei nomi predefinito

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

Utilizzo di segreti diversi per spazio dei nomi

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

Utilizzo di segreti diversi per volume

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb` Filtri per pool che supportano volumi SMB. `nasType: nfs` oppure `nasType: null` Filtri per i pool NFS.

Creare il backend

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 Cloud Volumes Service per il backend di Google Cloud

Scopri come configurare NetApp Cloud Volumes Service per Google Cloud come back-end per la tua installazione Astra Trident utilizzando le configurazioni di esempio fornite.

Dettagli del driver di Google Cloud

Astra Trident offre a. `gcp-cvs` driver per comunicare con il quadro strumenti. Le modalità di accesso supportate sono: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
`gcp-cvs`	NFS	Filesystem	RWO, ROX, RWX, RWOP	`nfs`

Scopri di più sul supporto di Astra Trident per Cloud Volumes Service per Google Cloud

Astra Trident può creare volumi Cloud Volumes Service in uno dei due "tipi di servizio":

- **CVS-Performance**: Il tipo di servizio Astra Trident predefinito. Questo tipo di servizio ottimizzato per le performance è più adatto per i carichi di lavoro di produzione che apprezzano le performance. Il tipo di servizio CVS-Performance è un'opzione hardware che supporta volumi con una dimensione minima di 100 GiB. È possibile scegliere tra "tre livelli di servizio":
  - `standard`
  - `premium`
  - `extreme`
- **CVS**: Il tipo di servizio CVS offre un'elevata disponibilità zonale con livelli di performance da limitati a moderati. Il tipo di servizio CVS è un'opzione software che utilizza pool di storage per supportare volumi di

dimensioni pari a 1 GiB. Il pool di storage può contenere fino a 50 volumi in cui tutti i volumi condividono la capacità e le performance del pool. È possibile scegliere tra "due livelli di servizio":

- `standardsw`
- `zoneredundantstandardsw`

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 Cloud Volumes Service
- Numero di progetto dell'account Google Cloud
- Account di servizio Google Cloud con `netappcloudvolumes.admin` ruolo
- File delle chiavi API per l'account Cloud Volumes Service

Opzioni di configurazione back-end

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.

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di storage	"gcp-cvs"
`backendName`	Nome personalizzato o backend dello storage	Nome del driver + "_" + parte della chiave API
`storageClass`	Parametro facoltativo utilizzato per specificare il tipo di servizio CVS. Utilizzare `software` Per selezionare il tipo di servizio CVS. In caso contrario, Astra Trident presuppone il tipo di servizio CVS-Performance (`hardware`).	
`storagePools`	Solo tipo di servizio CVS. Parametro facoltativo utilizzato per specificare i pool di storage per la creazione di volumi.	
`projectNumber`	Numero di progetto dell'account Google Cloud. Il valore si trova nella home page del portale Google Cloud.	
`hostProjectNumber`	Necessario se si utilizza una rete VPC condivisa. In questo scenario, `projectNumber` è il progetto di servizio, e. `hostProjectNumber` è il progetto host.	
`apiRegion`	La regione di Google Cloud in cui Astra Trident crea volumi Cloud Volumes Service. Quando si creano cluster Kubernetes con più aree, i volumi creati in un `apiRegion` Può essere utilizzato nei carichi di lavoro pianificati su nodi in più aree di Google Cloud. Il traffico interregionale comporta un costo aggiuntivo.	

Parametro	Descrizione	Predefinito
`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`	Livello di servizio CVS-Performance o CVS per i nuovi volumi. I valori CVS-Performance sono `standard`, `premium`, o. `extreme`. I valori CVS sono `standardsw` oppure `zoneredundantstandardsw`.	CVS-Performance (prestazioni CVS) è "standard". Il valore predefinito di CVS è "standardsw".
`network`	Rete Google Cloud utilizzata per i volumi Cloud Volumes Service.	"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

`allowedTopologies`	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`	

Opzioni di provisioning dei volumi

È possibile controllare il provisioning del volume predefinito in `defaults` del file di configurazione.

Parametro	Descrizione	Predefinito
`exportRule`	Le 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`	Accesso a `.snapshot` directory	"falso"
`snapshotReserve`	Percentuale di volume riservato agli snapshot	"" (accettare CVS come valore predefinito 0)
`size`	Le dimensioni dei nuovi volumi. Performance CVS minima: 100 GiB. CVS minimo: 1 GiB.	Per impostazione predefinita, il tipo di servizio CVS-Performance è "100GiB". Il tipo di servizio CVS non imposta un valore predefinito, ma richiede un minimo di 1 GiB.

Esempi di tipo di servizio CVS-Performance

I seguenti esempi forniscono configurazioni di esempio per il tipo di servizio CVS-Performance.

Esempio 1: Configurazione minima

Questa è la configurazione di backend minima che utilizza il tipo di servizio CVS-Performance predefinito con il livello di servizio "standard" predefinito.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  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 livello di servizio

In questo esempio vengono illustrate le opzioni di configurazione back-end, inclusi il livello di servizio e i valori predefiniti del volume.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  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 3: Configurazione del pool virtuale

Questo esempio utilizza `storage` per configurare i pool virtuali e il `StorageClasses` che fanno riferimento a loro. Fare riferimento a. Definizioni delle classi di storage per vedere come sono state definite le classi di storage.

In questo caso, vengono impostati valori predefiniti specifici per tutti i pool virtuali, che impostano `snapshotReserve` al 5% e a. `exportRule` a 0.0.0.0/0. I pool virtuali sono definiti in `storage` sezione. Ogni singolo pool virtuale definisce il proprio `serviceLevel`e alcuni pool sovrascrivono i valori predefiniti. Le etichette dei pool virtuali sono state utilizzate per differenziare i pool in base a. `performance` e. `protection`.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
```

```
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  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
```

Definizioni delle classi di storage

Le seguenti definizioni di StorageClass si applicano all'esempio di configurazione del pool virtuale. Utilizzo di `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.

Esempio di classe di storage

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

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

- Il primo StorageClass (`cvs-extreme-extra-protection`) viene mappato al primo pool 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 virtuale è selezionato e garantisce che il requisito di riserva snapshot sia soddisfatto.

Esempi di tipo di servizio CVS

I seguenti esempi forniscono configurazioni di esempio per il tipo di servizio CVS.

Esempio 1: Configurazione minima

Questa è la configurazione backend minima utilizzata `storageClass` Per specificare il tipo di servizio CVS e il valore predefinito `standardsw` livello di servizio.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
storageClass: software
apiRegion: us-east4
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  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
serviceLevel: standardsw
```

Esempio 2: Configurazione del pool di storage

Questo esempio di configurazione di backend utilizza `storagePools` per configurare un pool di storage.

```
---
version: 1
storageDriverName: gcp-cvs
backendName: gcp-std-so-with-pool
projectNumber: '531265380079'
apiRegion: europe-west1
apiKey:
  type: service_account
  project_id: cloud-native-data
  private_key_id: "<id_value>"
  private_key: |-
    -----BEGIN PRIVATE KEY-----
    MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQDaT+Oui9FBAw19
    L1AGEkrYU5xd9K5NlO5jMkIFND5wCD+Nv+jd1GvtFRLaLK5RvXyF5wzvztmODNS+
    qtScpQ+5cFpQkuGtv9U9+N6qtuVYYO3b504Kp5CtqVPJCgMJaK2j8pZTIqUiMum/
    5/Y9oTbZrjAHSMgJm2nHzFq2X0rqVMaHghI6ATm4DOuWx8XGWKTGIPlc0qPqJlqS
    LLaWOH4VIZQZCAyW5IUp9CAmwqHgdG0uhFNfCgMmED6PBUvVLsLvcq86X+QSWR9k
    ETqElj/sGCenPF7ti1DhGBFafd9hPnxg9PZY29ArEZwY9G/ZjZQX7WPgs0VvxiNR
    DxZRC3GXAgMBAAECggEACn5c59bG/qnVEVI1CwMAalM5M2z09JFhlLlljKwntNPj
    Vilw2eTW2+UE7HbJru/S7KQgA5Dnn9kvCraEahPRuddUMrD0vG4kTl/IODV6uFuk
    Y0sZfbqd4jMUQ21smvGsqFzwloYWS5qzO1W83ivXH/HW/iqkmY2eW+EPRS/hwSSu
    SscR+SojI7PB0BWSJhlV4yqYf3vcD/D95el2CVHfRCkL85DKumeZ+yHEnpiXGZAE
    t8xSs4a5OOPm6NHhevCw2a/UQ95/foXNUR450HtbjieJo5o+FF6EYZQGfU2ZHZO8
    37FBKuaJkdGW5xqaI9TL7aqkGkFMF4F2qvOZM+vy8QKBgQD4oVuOkJDlhkTHP86W
    esFlw1kpWyJR9ZA7LI0g/rVpslnX+XdDq0WQf4umdLNau5hYEH9LU6ZSGs1Xk3/B
    NHwR6OXFuqEKNiu83d0zSlHhTy7PZpOZdj5a/vVvQfPDMz7OvsqLRd7YCAbdzuQ0
    +Ahq0Ztwvg0HQ64hdW0ukpYRRwKBgQDgyHj98oqswoYuIa+pP1yS0pPwLmjwKyNm
    /HayzCp+Qjiyy7Tzg8AUqlH1Ou83XbV428jvg7kDhO7PCCKFq+mMmfqHmTpb0Maq
    KpKnZg4ipsqP1yHNNEoRmcailXbwIhCLewMqMrggUiLOmCw4PscL5nK+4GKu2XE1
    jLqjWAZFMQKBgFHkQ9XXRAJ1kR3XpGHoGN890pZOkCVSrqju6aUef/5KYlFCt8ew
    F/+aIxM2iQSvmWQYOvVCnhuY/F2GFaQ7d0om3decuwI0CX/xy7PjHMkLXa2uaZs4
    WR17sLduj62RqXRLX0c0QkwBiNFyHbRcpdkZJQujbYMhBa+7j7SxT4BtAoGAWMWT
    UucocRXZm/pdvz9wteNH3YDWnJLMxm1KC06qMXbBoYrliY4sm3ywJWMC+iCd/H8A
    Gecxd/xVu5mA2L2N3KMq18Zhz8Th0G5DwKyDRJgOQ0Q46yuNXOoYEjlo4Wjyk8Me
    +tlQ8iK98E0UmZnhTgfSpSNElbz2AqnzQ3MN9uECgYAqdvdVPnKGfvdtZ2DjyMoJ
    E89UIC41WjjJGmHsd8W65+3X0RwMzKMT6aZc5tK9J5dHvmWIETnbM+lTImdBBFga
    NWOC6f3r2xbGXHhaWSl+nobpTuvlo56ZRJVvVk7lFMsiddzMuHH8pxfgNJemwA4P
    ThDHCejv035NNV6KyoO0tA==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@cloud-native-
data.iam.gserviceaccount.com
  client_id: '107071413297115343396'
```

```
  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%40cloud-native-data.iam.gserviceaccount.com
storageClass: software
zone: europe-west1-b
network: default
storagePools:
- 1bc7f380-3314-6005-45e9-c7dc8c2d7509
serviceLevel: Standardsw
```

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 di elementi con la tua installazione Astra Trident.

Dettagli driver elemento

Astra Trident offre a. `solidfire-san` driver di archiviazione per comunicare con il cluster. Le modalità di accesso supportate sono: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Il `solidfire-san` il driver di archiviazione supporta 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, RWOP	Nessun filesystem. Dispositivo a blocchi raw.
`solidfire-san`	ISCSI	Filesystem	RWO, RWOP	`xfs`, `ext3`, `ext4`

Prima di iniziare

Prima di creare un backend elemento, è necessario quanto segue.

- 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. Fare riferimento a. "informazioni sulla preparazione del nodo di lavoro".

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. Astra Trident utilizza il protocollo CHAP.	vero
`AccessGroups`	Elenco degli ID del gruppo di accesso da utilizzare	Trova l'ID di un gruppo di accesso denominato "tridente"
`Types`	Specifiche QoS	

Parametro	Descrizione	Predefinito
`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.

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 virtuali

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

Astra Trident copia le etichette presenti su un pool di storage nel LUN dello storage back-end al momento del provisioning. Per comodità, gli amministratori dello storage possono definire le etichette per ogni pool virtuale e raggruppare i volumi per etichetta.

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 virtuali sono definiti in `storage` sezione. In questo esempio, alcuni pool di storage impostano il proprio tipo e alcuni pool sovrascrivono i valori predefiniti impostati in precedenza.

```
---
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 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 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 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"

Driver SAN ONTAP

Panoramica del driver SAN ONTAP

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

Dettagli del driver SAN ONTAP

Astra Trident offre i seguenti driver per lo storage SAN per comunicare con il cluster ONTAP. Le modalità di accesso supportate sono: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Se stai utilizzando Astra Control per protezione, recovery e mobilità, leggi Compatibilità driver Astra Control.

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
`ontap-san`	ISCSI	Blocco	RWO, ROX, RWX, RWOP	Nessun file system; dispositivo a blocchi raw
`ontap-san`	ISCSI	Filesystem	RWO, RWOP ROX e RWX non sono disponibili in modalità Volume filesystem.	`xfs`, `ext3`, `ext4`
`ontap-san`	NVMe/TCP Fare riferimento a. Considerazioni aggiuntive su NVMe/TCP.	Blocco	RWO, ROX, RWX, RWOP	Nessun file system; dispositivo a blocchi raw

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
`ontap-san`	NVMe/TCP Fare riferimento a. Considerazioni aggiuntive su NVMe/TCP.	Filesystem	RWO, RWOP ROX e RWX non sono disponibili in modalità Volume filesystem.	`xfs`, `ext3`, `ext4`
`ontap-san-economy`	ISCSI	Blocco	RWO, ROX, RWX, RWOP	Nessun file system; dispositivo a blocchi raw
`ontap-san-economy`	ISCSI	Filesystem	RWO, RWOP ROX e RWX non sono disponibili in modalità Volume filesystem.	`xfs`, `ext3`, `ext4`

Compatibilità driver Astra Control

Astra Control offre protezione perfetta, disaster recovery e mobilità (spostamento di volumi tra cluster Kubernetes) per i volumi creati con `ontap-nas`, `ontap-nas-flexgroup`, e. `ontap-san` driver. Fare riferimento a. "Prerequisiti per la replica di Astra Control" per ulteriori informazioni.

- Utilizzare `ontap-san-economy` solo se si prevede che il conteggio dell'utilizzo persistente del volume sia superiore a. "Limiti di volume ONTAP supportati".
- Utilizzare `ontap-nas-economy` solo se si prevede che il conteggio dell'utilizzo persistente del volume sia superiore a. "Limiti di volume ONTAP supportati" e a. `ontap-san-economy` impossibile utilizzare il driver.
- Non utilizzare `ontap-nas-economy` se prevedete la necessità di protezione dei dati, disaster recovery o mobilità.

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.

Considerazioni aggiuntive su NVMe/TCP

Astra Trident supporta il protocollo non-volatile memory express (NVMe) utilizzando il `ontap-san` driver che include:

- IPv6
- Snapshot e cloni di volumi NVMe
- Ridimensionamento di un volume NVMe
- Importare un volume NVMe creato al di fuori di Astra Trident in modo che il suo ciclo di vita possa essere gestito da Astra Trident
- Multipath nativo NVMe
- Arresto anomalo o anomalo dei K8s nodi (24,02)

Astra Trident non supporta:

- DH-HMAC-CHAP supportato nativamente da NVMe
- Multipathing DM (Device mapper)
- Crittografia LUKS

Prepararsi a configurare il backend con i driver SAN ONTAP

Comprendere i requisiti e le opzioni di autenticazione per la configurazione di un backend ONTAP con i driver SAN ONTAP.

Requisiti

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. Fare riferimento a. "Preparare il nodo di lavoro" per ulteriori informazioni.

Autenticare il backend ONTAP

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).

È possibile aggiornare i backend esistenti per passare da un metodo basato su credenziali a un metodo

basato su certificato. Tuttavia, è supportato un solo metodo di autenticazione alla volta. Per passare a un metodo di autenticazione diverso, è necessario rimuovere il metodo esistente dalla configurazione di back-end.

> Se si tenta di fornire **credenziali e certificati**, la creazione del backend non riesce e viene visualizzato un errore che indica che nel file di configurazione sono stati forniti più metodi di autenticazione.

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:

YAML

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

JSON

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

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 o l'aggiornamento 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",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"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, è necessario rimuovere il metodo di autenticazione esistente e aggiungere il nuovo metodo di autenticazione. Quindi, utilizzare il file backend.json aggiornato contenente i parametri necessari per l'esecuzione `tridentctl backend update`.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"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.

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 sicurezza dell'iniziatore predefinito della SVM su CHAP bidirezionale e imposta il nome utente e i segreti dal 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: password
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 iniziatore predefinito della SVM è nessuno (impostato per impostazione predefinita) **e** nel volume non sono già presenti LUN preesistenti, 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. In questo modo, l'accesso ai LUN già presenti nella SVM non è 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).

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": "password",
    "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 ed esempi di configurazione DELLA SAN ONTAP

Scopri come creare e utilizzare i driver SAN ONTAP con la tua installazione Astra Trident. In questa sezione vengono forniti esempi di configurazione backend e dettagli per la mappatura dei 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 di una LIF di gestione SVM. È possibile specificare un nome di dominio completo (FQDN). Può essere impostato per utilizzare gli indirizzi IPv6 se Astra Trident è stato installato utilizzando il flag IPv6. Gli indirizzi IPv6 devono essere definiti tra parentesi quadre, ad esempio `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Per uno switchover perfetto del MetroCluster, consulta il Esempio MetroCluster.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Indirizzo IP del protocollo LIF. **Non specificare iSCSI.** utilizza Astra Trident "Mappa LUN selettiva ONTAP" Per scoprire i LIF iSCI necessari per stabilire una sessione multi-percorso. Viene generato un avviso se `dataLIF` è esplicitamente definito. **Ometti per MetroCluster.** vedere la Esempio MetroCluster.	Derivato dalla SVM
`svm`	Macchina virtuale per lo storage da utilizzare **Ometti per MetroCluster.** vedere la Esempio MetroCluster.	Derivato se un SVM `managementLIF` è specificato
`useCHAP`	Utilizzare CHAP per autenticare iSCSI per i driver SAN ONTAP [booleano]. Impostare su `true` Affinché Astra Trident configuri e utilizzi CHAP bidirezionale come autenticazione predefinita per la SVM fornita nel backend. Fare riferimento a. "Prepararsi a configurare il backend con i driver SAN ONTAP" per ulteriori informazioni.	`false`
`chapInitiatorSecret`	Segreto iniziatore CHAP. Necessario se `useCHAP=true`	""
`labels`	Set di etichette arbitrarie formattate con JSON da applicare sui volumi	""

Parametro	Descrizione	Predefinito
`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	""
`trustedCACertificate`	Valore codificato in base64 del certificato CA attendibile. Opzionale. Utilizzato per l'autenticazione basata su certificato.	""
`username`	Nome utente necessario per comunicare con il cluster ONTAP. Utilizzato per l'autenticazione basata su credenziali.	""
`password`	Password necessaria per comunicare con il cluster ONTAP. Utilizzato per l'autenticazione basata su credenziali.	""
`svm`	Macchina virtuale per lo storage da utilizzare	Derivato se un SVM `managementLIF` è specificato
`storagePrefix`	Prefisso utilizzato per il provisioning di nuovi volumi nella SVM. Non può essere modificato in seguito. Per aggiornare questo parametro, è necessario creare un nuovo backend.	`trident`
`limitAggregateUsage`	Il provisioning non riesce se l'utilizzo è superiore a questa percentuale. Se si utilizza un backend Amazon FSX per NetApp ONTAP, non specificare `limitAggregateUsage`. Il fornito `fsxadmin` e. `vsadmin` Non includere le autorizzazioni necessarie per recuperare l'utilizzo aggregato e limitarlo utilizzando Astra Trident.	"" (non applicato per impostazione predefinita)
`limitVolumeSize`	Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore. Limita inoltre le dimensioni massime dei volumi gestiti per qtree e LUN.	"" (non applicato per impostazione predefinita)
`lunsPerFlexvol`	LUN massimi per FlexVol, devono essere compresi nell'intervallo [50, 200]	`100`

Parametro	Descrizione	Predefinito
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Esempio, {"api":false, "method":true} Non utilizzare a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.	`null`
`useREST`	Parametro booleano per l'utilizzo delle API REST di ONTAP. **Anteprima tecnica** `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 9.11.1 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. `useREST` Non è supportato con MetroCluster. `useREST` È pienamente qualificato per NVMe/TCP.	`false`
`sanType`	Utilizzare per selezionare `iscsi` Per iSCSI o. `nvme` Per NVMe/TCP.	`iscsi` se vuoto

Opzioni di configurazione back-end per il provisioning dei volumi

È possibile controllare il provisioning predefinito utilizzando queste opzioni in `defaults` 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 prenotazione dello spazio; "nessuno" (sottile) o "volume" (spesso)	"nessuno"
`snapshotPolicy`	Policy di Snapshot da utilizzare	"nessuno"

Parametro	Descrizione	Predefinito
`qosPolicy`	Gruppo di criteri QoS da assegnare per i volumi creati. Scegliere tra qosPolicy o adaptiveQosPolicy per pool di storage/backend. L'utilizzo di gruppi di policy QoS con Astra Trident richiede ONTAP 9.8 o versione successiva. Si consiglia di utilizzare un gruppo di policy QoS non condiviso e di assicurarsi che il gruppo di policy venga 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.	""
`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"
`encryption`	Abilitare NetApp Volume Encryption (NVE) sul nuovo volume; il valore predefinito è `false`. NVE deve essere concesso in licenza e abilitato sul cluster per utilizzare questa opzione. Se NAE è attivato sul backend, tutti i volumi forniti in Astra Trident saranno abilitati per NAE. Per ulteriori informazioni, fare riferimento a: "Come funziona Astra Trident con NVE e NAE".	"falso"
`luksEncryption`	Attivare la crittografia LUKS. Fare riferimento a. "Utilizzo di Linux Unified Key Setup (LUKS)". La crittografia LUKS non è supportata per NVMe/TCP.	""
`securityStyle`	Stile di sicurezza per nuovi volumi	`unix`
`tieringPolicy`	Criterio di tiering da utilizzare "nessuno"	"Solo Snapshot" per la configurazione SVM-DR pre-ONTAP 9,5

Esempi di provisioning di volumi

Ecco un esempio con i valori predefiniti definiti:

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
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:

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.

Esempio DI SAN ONTAP

Si tratta di una configurazione di base che utilizza `ontap-san` driver.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

Esempio di economia SAN ONTAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
username: vsadmin
password: <password>
```

Esempio MetroCluster

È possibile configurare il backend per evitare di dover aggiornare manualmente la definizione del backend dopo lo switchover e lo switchback durante "Replica e recovery di SVM".

Per uno switchover e uno switchback perfetto, specifica la SVM utilizzando `managementLIF` e omettere `dataLIF` e. `svm` parametri. Ad esempio:

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

Esempio di autenticazione basata su certificato

In questo esempio di configurazione di base `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
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

Esempi CHAP bidirezionali

Questi esempi creano un backend con `useCHAP` impostare su `true`.

Esempio di SAN ONTAP CHAP

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

Esempio di ONTAP SAN economy CHAP

```
---
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
username: vsadmin
password: <password>
```

Esempio NVMe/TCP

Devi disporre di una SVM configurata con NVMe sul back-end ONTAP. Si tratta di una configurazione backend di base per NVMe/TCP.

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

Esempi di backend con pool virtuali

In questi file di definizione back-end di esempio, 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 virtuali sono definiti nella sezione storage.

Astra Trident imposta le etichette di provisioning nel campo "commenti". I commenti vengono impostati su FlexVol. Astra Trident copia tutte le etichette presenti su un pool virtuale nel volume di storage al momento del provisioning. Per comodità, gli amministratori dello storage possono definire le etichette per ogni pool virtuale e raggruppare i volumi per etichetta.

In questi esempi, alcuni dei pool di storage sono impostati in modo personalizzato `spaceReserve`, `spaceAllocation`, e. `encryption` e alcuni pool sovrascrivono i valori predefiniti.

Esempio DI SAN ONTAP

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
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'
```

Esempio di economia SAN ONTAP

```
---
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
username: vsadmin
password: <password>
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'
- labels:
    department: legal
    creditpoints: '5000'
  zone: us_east_1c
```

```
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
```

Esempio NVMe/TCP

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: 'false'
  encryption: 'true'
storage:
- labels:
    app: testApp
    cost: '20'
  defaults:
    spaceAllocation: 'false'
    encryption: 'false'
```

Mappare i backend in StorageClasses

Le seguenti definizioni di StorageClass fanno riferimento a. Esempi di backend con pool virtuali. 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 `protection-gold` StorageClass verrà mappato al primo pool virtuale in `ontap-san` back-end. Questo è l'unico pool che offre una protezione di livello gold.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- Il `protection-not-gold` StorageClass eseguirà il mapping al secondo e al terzo pool virtuale in `ontap-san` back-end. Questi sono gli unici pool che offrono un livello di protezione diverso dall'oro.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- Il `app-mysqldb` StorageClass eseguirà il mapping al terzo pool virtuale in `ontap-san-economy` back-end. Questo è l'unico pool che offre la configurazione del pool di storage per l'applicazione di tipo mysqldb.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- Il `protection-silver-creditpoints-20k` StorageClass eseguirà il mapping al secondo pool virtuale in `ontap-san` back-end. Questo è l'unico pool che offre una protezione di livello Silver e 20000 punti di credito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- Il `creditpoints-5k` StorageClass eseguirà il mapping al terzo pool virtuale in `ontap-san` il back-end e il quarto pool virtuale in `ontap-san-economy` back-end. Queste sono le uniche offerte di pool con 5000 punti di credito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

- Il `my-test-app-sc` StorageClass verrà mappato su `testAPP` pool virtuale in `ontap-san` conducente con `sanType: nvme`. Si tratta dell'unica offerta di piscina `testApp`.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"
```

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

Driver NAS ONTAP

Panoramica del driver NAS ONTAP

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

Dettagli del driver NAS ONTAP

Astra Trident offre i seguenti driver di storage NAS per comunicare con il cluster ONTAP. Le modalità di accesso supportate sono: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Se stai utilizzando Astra Control per protezione, recovery e mobilità, leggi Compatibilità driver Astra Control.

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
`ontap-nas`	NFS PMI	Filesystem	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
`ontap-nas-economy`	NFS PMI	Filesystem	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`
`ontap-nas-flexgroup`	NFS PMI	Filesystem	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

Compatibilità driver Astra Control

Astra Control offre protezione perfetta, disaster recovery e mobilità (spostamento di volumi tra cluster Kubernetes) per i volumi creati con `ontap-nas`, `ontap-nas-flexgroup`, e. `ontap-san` driver. Fare riferimento a. "Prerequisiti per la replica di Astra Control" per ulteriori informazioni.

- Utilizzare `ontap-san-economy` solo se si prevede che il conteggio dell'utilizzo persistente del volume sia superiore a. "Limiti di volume ONTAP supportati".
- Utilizzare `ontap-nas-economy` solo se si prevede che il conteggio dell'utilizzo persistente del volume sia superiore a. "Limiti di volume ONTAP supportati" e a. `ontap-san-economy` impossibile utilizzare il driver.
- Non utilizzare `ontap-nas-economy` se prevedete la necessità di protezione dei dati, disaster recovery o mobilità.

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.

Prepararsi a configurare un backend con i driver NAS ONTAP

Comprendere i requisiti, le opzioni di autenticazione e le policy di esportazione per la configurazione di un backend ONTAP con i driver NAS ONTAP.

Requisiti

- Per tutti i backend ONTAP, Astra Trident richiede almeno un aggregato assegnato alla SVM.
- È possibile eseguire più di un driver e creare classi di storage che puntano 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. Fare riferimento a. "qui" per ulteriori dettagli.
- Astra Trident supporta volumi SMB montati su pod eseguiti solo su nodi Windows. Fare riferimento a. Preparatevi al provisioning dei volumi SMB per ulteriori informazioni.

Autenticare il backend ONTAP

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

- Basato sulle credenziali: Questa modalità richiede autorizzazioni sufficienti per il backend ONTAP. Si consiglia di utilizzare un account associato a 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: Questa modalità richiede un certificato installato sul backend affinché Astra Trident possa comunicare con un cluster ONTAP. 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).

È possibile aggiornare i backend esistenti per passare da un metodo basato su credenziali a un metodo basato su certificato. Tuttavia, è supportato un solo metodo di autenticazione alla volta. Per passare a un metodo di autenticazione diverso, è necessario rimuovere il metodo esistente dalla configurazione di back-end.

> Se si tenta di fornire **credenziali e certificati**, la creazione del backend non riesce e viene visualizzato un errore che indica che nel file di configurazione sono stati forniti più metodi di autenticazione.

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:

YAML

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

JSON

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

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, è necessario rimuovere il metodo di autenticazione esistente e aggiungere il nuovo metodo di autenticazione. Quindi, utilizzare il file backend.json aggiornato contenente i parametri necessari per l'esecuzione `tridentctl update backend`.

```
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": "password",
"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

Astra Trident permette di gestire in modo dinamico 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.

> Non utilizzare NAT (Network Address Translation) quando si utilizzano criteri di esportazione dinamici. Con NAT, il controller di archiviazione rileva l'indirizzo NAT di frontend e non l'indirizzo host IP effettivo, pertanto l'accesso viene negato quando non viene trovata alcuna corrispondenza nelle regole di esportazione.

Esempio

È necessario utilizzare due opzioni di configurazione. Ecco un esempio di definizione di backend:

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
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 creato in precedenza con una regola di esportazione che consenta il blocco CIDR del nodo (ad esempio il criterio di esportazione predefinito). Segui sempre le Best practice consigliate da NetApp per dedicare una SVM a 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 verrà creato un nuovo criterio di esportazione denominato dopo l'UUID del backend e i volumi presenti sul backend utilizzeranno il

criterio di esportazione appena creato quando vengono nuovamente montati.

> 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.

Preparatevi al provisioning dei volumi SMB

Con un po' di preparazione aggiuntiva, puoi eseguire il provisioning dei volumi SMB utilizzando `ontap-nas` driver.

> Per creare un, è necessario configurare entrambi i protocolli NFS e SMB/CIFS su SVM `ontap-nas-economy` Volume SMB per ONTAP on-premise. La mancata configurazione di uno di questi protocolli causerà un errore nella creazione del volume SMB.

Prima di iniziare

Prima di eseguire il provisioning di volumi SMB, è necessario disporre di quanto segue.

- Un cluster Kubernetes con un nodo controller Linux e almeno un nodo di lavoro Windows che esegue Windows Server 2019. Astra Trident supporta volumi SMB montati su pod eseguiti solo su nodi Windows.
- Almeno un segreto Astra Trident contenente le credenziali Active Directory. Per generare un segreto `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Proxy CSI configurato come servizio Windows. Per configurare un `csi-proxy`, fare riferimento a. "GitHub: Proxy CSI" oppure "GitHub: Proxy CSI per Windows" Per i nodi Kubernetes in esecuzione su Windows.

Fasi

1. Per ONTAP on-premise, è possibile creare una condivisione SMB oppure Astra Trident ne può creare una per te.

   > Le condivisioni SMB sono richieste per Amazon FSX per ONTAP.

   È possibile creare le condivisioni amministrative SMB in due modi utilizzando "Console di gestione Microsoft" Snap-in cartelle condivise o utilizzo dell'interfaccia CLI di ONTAP. Per creare le condivisioni SMB utilizzando la CLI ONTAP:

   a. Se necessario, creare la struttura del percorso di directory per la condivisione.

      Il `vserver cifs share create` il comando controlla il percorso specificato nell'opzione -path durante la creazione della condivisione. Se il percorso specificato non esiste, il comando non riesce.

   b. Creare una condivisione SMB associata alla SVM specificata:

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

c. Verificare che la condivisione sia stata creata:

```
vserver cifs share show -share-name share_name
```

Fare riferimento a. "Creare una condivisione SMB" per informazioni dettagliate.

2. Quando si crea il backend, è necessario configurare quanto segue per specificare i volumi SMB. Per tutte le opzioni di configurazione backend FSX per ONTAP, fare riferimento a. "FSX per le opzioni di configurazione e gli esempi di ONTAP".

Parametro	Descrizione	Esempio
`smbShare`	È possibile specificare una delle seguenti opzioni: Il nome di una condivisione SMB creata utilizzando la console di gestione Microsoft o l'interfaccia utente di ONTAP; un nome per consentire ad Astra Trident di creare la condivisione SMB; oppure è possibile lasciare vuoto il parametro per impedire l'accesso condiviso ai volumi. Questo parametro è facoltativo per ONTAP on-premise. Questo parametro è obbligatorio per i backend Amazon FSX per ONTAP e non può essere vuoto.	`smb-share`
`nasType`	**Deve essere impostato su `smb`.** se null, il valore predefinito è `nfs`.	`smb`
`securityStyle`	Stile di sicurezza per nuovi volumi. **Deve essere impostato su `ntfs` oppure `mixed` Per volumi SMB.**	`ntfs` oppure `mixed` Per volumi SMB
`unixPermissions`	Per i nuovi volumi. **Deve essere lasciato vuoto per i volumi SMB.**	""

Opzioni ed esempi di configurazione del NAS ONTAP

Scopri come creare e utilizzare i driver NAS ONTAP con la tua installazione Astra Trident. In questa sezione vengono forniti esempi di configurazione backend e dettagli per la mappatura dei 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 È possibile specificare un nome di dominio completo (FQDN). Può essere impostato per utilizzare gli indirizzi IPv6 se Astra Trident è stato installato utilizzando il flag IPv6. Gli indirizzi IPv6 devono essere definiti tra parentesi quadre, ad esempio `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Per uno switchover perfetto del MetroCluster, consulta il Esempio MetroCluster.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Indirizzo IP del protocollo LIF. Si consiglia di specificare `dataLIF`. Se non fornito, Astra Trident recupera i dati LIF dalla SVM. È possibile specificare un FQDN (Fully-qualified domain name) da utilizzare per le operazioni di montaggio NFS, consentendo di creare un DNS round-robin per il bilanciamento del carico tra più LIF di dati. Può essere modificato dopo l'impostazione iniziale. Fare riferimento a. . Può essere impostato per utilizzare gli indirizzi IPv6 se Astra Trident è stato installato utilizzando il flag IPv6. Gli indirizzi IPv6 devono essere definiti tra parentesi quadre, ad esempio `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **Ometti per MetroCluster.** vedere la Esempio MetroCluster.	Indirizzo specificato o derivato da SVM, se non specificato (non consigliato)
`svm`	Macchina virtuale per lo storage da utilizzare **Ometti per MetroCluster.** vedere la Esempio MetroCluster.	Derivato se un SVM `managementLIF` è specificato

Parametro	Descrizione	Predefinito
`autoExportPolicy`	Abilita la creazione e l'aggiornamento automatici dei criteri di esportazione [booleano]. Utilizzando il `autoExportPolicy` e. `autoExportCIDRs` Astra Trident è in grado di gestire automaticamente le policy di esportazione.	falso
`autoExportCIDRs`	Elenco di CIDR per filtrare gli IP dei nodi di Kubernetes rispetto a quando `autoExportPolicy` è attivato. Utilizzando il `autoExportPolicy` e. `autoExportCIDRs` Astra Trident è in grado di gestire automaticamente le policy di esportazione.	["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	""
`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	
`storagePrefix`	Prefisso utilizzato per il provisioning di nuovi volumi nella SVM. Non può essere aggiornato dopo l'impostazione	"trident"
`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. Inoltre, limita le dimensioni massime dei volumi gestiti per qtree e LUN, oltre a `qtreesPerFlexvol` Consente di personalizzare il numero massimo di qtree per FlexVol.	"" (non applicato per impostazione predefinita)
`lunsPerFlexvol`	LUN massimi per FlexVol, devono essere compresi nell'intervallo [50, 200]	"100"

Parametro	Descrizione	Predefinito
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Esempio, {"api":false, "method":true} Non utilizzare `debugTraceFlags` a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.	nullo
`nasType`	Configurare la creazione di volumi NFS o SMB. Le opzioni sono `nfs`, `smb` o nullo. L'impostazione su Null consente di impostare i volumi NFS come predefiniti.	`nfs`
`nfsMountOptions`	Elenco separato da virgole delle opzioni di montaggio NFS. 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.	""
`qtreesPerFlexvol`	Qtree massimi per FlexVol, devono essere compresi nell'intervallo [50, 300]	"200"
`smbShare`	È possibile specificare una delle seguenti opzioni: Il nome di una condivisione SMB creata utilizzando la console di gestione Microsoft o l'interfaccia utente di ONTAP; un nome per consentire ad Astra Trident di creare la condivisione SMB; oppure è possibile lasciare vuoto il parametro per impedire l'accesso condiviso ai volumi. Questo parametro è facoltativo per ONTAP on-premise. Questo parametro è obbligatorio per i backend Amazon FSX per ONTAP e non può essere vuoto.	`smb-share`

Parametro	Descrizione	Predefinito
`useREST`	Parametro booleano per l'utilizzo delle API REST di ONTAP. **Anteprima tecnica** `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 9.11.1 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. `useREST` Non è supportato con MetroCluster.	falso

Opzioni di configurazione back-end per il provisioning dei volumi

È possibile controllare il provisioning predefinito utilizzando queste opzioni in `defaults` 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 prenotazione 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"

Parametro	Descrizione	Predefinito
`encryption`	Abilitare NetApp Volume Encryption (NVE) sul nuovo volume; il valore predefinito è `false`. NVE deve essere concesso in licenza e abilitato sul cluster per utilizzare questa opzione. Se NAE è attivato sul backend, tutti i volumi forniti in Astra Trident saranno abilitati per NAE. Per ulteriori informazioni, fare riferimento a: "Come funziona Astra Trident con NVE e NAE".	"falso"
`tieringPolicy`	Criterio di tiering da utilizzare "nessuno"	"Solo Snapshot" per la configurazione SVM-DR pre-ONTAP 9,5
`unixPermissions`	Per i nuovi volumi	"777" per i volumi NFS; vuoto (non applicabile) per i volumi SMB
`snapshotDir`	Controlla l'accesso a. `.snapshot` directory	"falso"
`exportPolicy`	Policy di esportazione da utilizzare	"predefinito"
`securityStyle`	Stile di sicurezza per nuovi volumi. Supporto di NFS `mixed` e. `unix` stili di sicurezza. Supporto SMB `mixed` e. `ntfs` stili di sicurezza.	Il valore predefinito di NFS è `unix`. Il valore predefinito di SMB è `ntfs`.

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.

Esempi di provisioning di volumi

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:

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.

Esempio di economia NAS ONTAP

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

Esempio di FlexGroup NAS ONTAP

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

Esempio MetroCluster

È possibile configurare il backend per evitare di dover aggiornare manualmente la definizione del backend dopo lo switchover e lo switchback durante "Replica e recovery di SVM".

Per uno switchover e uno switchback perfetto, specifica la SVM utilizzando `managementLIF` e omettere `dataLIF` e. `svm` parametri. Ad esempio:

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

Esempio di volumi SMB

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
nasType: smb
securityStyle: ntfs
unixPermissions: ""
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Esempio di 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_
```

Esempio di 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: password
nfsMountOptions: nfsvers=4
```

Esempio di indirizzi IPv6

Questo esempio mostra `managementLIF` Utilizzando un indirizzo 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: password
```

Esempio di Amazon FSX per ONTAP con volumi SMB

Il `smbShare` Il parametro è obbligatorio per FSX per ONTAP che utilizza volumi SMB.

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

Esempi di backend con pool virtuali

Nei file di definizione back-end di esempio illustrati 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 virtuali sono definiti nella sezione storage.

Astra Trident imposta le etichette di provisioning nel campo "commenti". I commenti sono impostati su FlexVol per `ontap-nas` O FlexGroup per `ontap-nas-flexgroup`. Astra Trident copia tutte le etichette presenti su un pool virtuale nel volume di storage al momento del provisioning. Per comodità, gli amministratori dello storage possono definire le etichette per ogni pool virtuale e raggruppare i volumi per etichetta.

In questi esempi, alcuni dei pool di storage sono impostati in modo personalizzato `spaceReserve`, `spaceAllocation`, e. `encryption` e alcuni pool sovrascrivono i valori predefiniti.

Esempio di NAS ONTAP

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
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:
    department: legal
    creditpoints: '5000'
  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'
```

Esempio di NAS FlexGroup ONTAP

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
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'
```

Esempio di economia NAS ONTAP

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
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:
    protection: bronze
    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 fanno riferimento a. Esempi di backend con pool virtuali. 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 `protection-gold` StorageClass eseguirà il mapping al primo e al secondo pool virtuale in `ontap-nas-flexgroup` back-end. Questi sono gli unici pool che offrono una protezione di livello gold.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- Il `protection-not-gold` StorageClass eseguirà il mapping al terzo e al quarto pool virtuale in `ontap-nas-flexgroup` back-end. Questi sono gli unici pool che offrono un livello di protezione diverso dall'oro.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- Il `app-mysqldb` StorageClass eseguirà il mapping al quarto pool virtuale in `ontap-nas` back-end. Questo è l'unico pool che offre la configurazione del pool di storage per l'applicazione di tipo mysqldb.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- Til `protection-silver-creditpoints-20k` StorageClass eseguirà il mapping al terzo pool virtuale in `ontap-nas-flexgroup` back-end. Questo è l'unico pool che offre una protezione di livello Silver e 20000 punti di credito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- Il `creditpoints-5k` StorageClass eseguirà il mapping al terzo pool virtuale in `ontap-nas` il back-end e il secondo pool virtuale in `ontap-nas-economy` back-end. Queste sono le uniche offerte di pool con 5000 punti di credito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

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

Aggiornare `dataLIF` dopo la configurazione iniziale

È possibile modificare la LIF dei dati dopo la configurazione iniziale eseguendo il seguente comando per fornire al nuovo file JSON di back-end i dati aggiornati LIF.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

Se i PVC sono collegati a uno o più pod, è necessario abbassare tutti i pod corrispondenti e riportarli di nuovo in alto per rendere effettiva la nuova LIF dei dati.

Amazon FSX per NetApp ONTAP

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 NetApp ONTAP. FSX per ONTAP 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 per ONTAP supporta le funzionalità del file system ONTAP e le API di amministrazione.

Panoramica

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.

Considerazioni

- Volumi SMB:
  - I volumi SMB sono supportati utilizzando `ontap-nas` solo driver.
  - I volumi SMB non sono supportati con il componente aggiuntivo Astra Trident EKS.
  - Astra Trident supporta volumi SMB montati su pod eseguiti solo su nodi Windows.
- Prima di Astra Trident 24,02, i volumi creati su file system Amazon FSX con backup automatici abilitati, non possono essere eliminati da Trident. Per evitare questo problema in Astra Trident 24,02 o versioni successive, specifica il `fsxFilesystemID`, AWS `apiRegion`, AWS `apikey`E AWS `secretKey` Nel file di configurazione back-end per AWS FSX per ONTAP.

Se stai specificando un ruolo IAM in Astra Trident, puoi omettere di specificare `apiRegion`, `apiKey`, e. `secretKey` Campi di Astra Trident esplicitamente. Per ulteriori informazioni, fare riferimento a. "FSX per le opzioni di configurazione e gli esempi di ONTAP".

Dettagli del driver FSX per ONTAP

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.

Per informazioni dettagliate sul conducente, fare riferimento a. "Driver NAS" e. "Driver SAN".

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.

> 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 consiglia vivamente di utilizzare `vsadmin` Con Astra Trident.

È possibile aggiornare i back-end per passare da un metodo basato su credenziali a un metodo basato su certificato. Tuttavia, se si tenta di fornire **credenziali e certificati**, la creazione del backend non riesce. Per passare a un metodo di autenticazione diverso, è necessario rimuovere il metodo esistente dalla configurazione di back-end.

Per ulteriori informazioni sull'attivazione dell'autenticazione, fare riferimento all'autenticazione per il tipo di driver in uso:

- "Autenticazione NAS ONTAP"
- "Autenticazione SAN ONTAP"

Identità cloud per EKS

L'identità del cloud consente ai pod Kubernetes di accedere alle risorse AWS eseguendo l'autenticazione come ruolo AWS IAM anziché fornire credenziali AWS esplicite.

Per sfruttare l'identità cloud in AWS, è necessario disporre di:

- Un cluster Kubernetes implementato utilizzando EKS
- Astra Trident ha installato che include `cloudProvider` specifica `"AWS"` e. `cloudIdentity` Specifica del ruolo AWS IAM.

Operatore Trident

Per installare Astra Trident usando l'operatore Trident, modifica `tridentorchestrator_cr.yaml` da impostare `cloudProvider` a. `"AWS"` e impostare `cloudIdentity` Al ruolo AWS IAM.

Ad esempio:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/astratrident-role'"
```

Timone

Impostare i valori per i flag **cloud provider** e **cloud Identity** utilizzando le seguenti variabili di ambiente:

```
export CP="AWS"
export CI="'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/astratrident-role'"
```

Nell'esempio seguente viene installato e impostato Astra Trident `cloudProvider` a. `AWS` utilizzo della variabile di ambiente `$CP` E imposta "cloudIdentity" utilizzando la variabile d'ambiente `$CI`:

```
helm install trident trident-operator-100.2402.0.tgz --set
cloudProvider=$CP --set cloudIdentity=$CI
```

<code>tridentctl</code>

Impostare i valori per i flag **cloud provider** e **cloud Identity** utilizzando le seguenti variabili di ambiente:

```
export CP="AWS"
export CI="'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/astratrident-role'"
```

Nell'esempio seguente viene installato Astra Trident e impostato l' `cloud-provider` contrassegna come `$CP`, e. `cloud-identity` a. `$CI`:

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

Trova ulteriori informazioni

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

Integra Amazon FSX per NetApp ONTAP

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

Requisiti

Oltre a "Requisiti di Astra Trident"Per integrare FSX per ONTAP con Astra Trident, sono necessari:

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

- Astra Trident supporta volumi SMB montati su pod eseguiti solo su nodi Windows. Fare riferimento a. Preparatevi al provisioning dei volumi SMB per ulteriori informazioni.

Integrazione dei driver ONTAP SAN e NAS

> Se si configurano volumi SMB, è necessario leggere Preparatevi al provisioning dei volumi SMB prima di creare il backend.

Fasi

1. Implementare Astra Trident utilizzando uno dei "metodi di implementazione".
2. Raccogliere il nome DNS LIF di gestione 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 "Autenticazione backend NAS" oppure "Autenticazione back-end SAN".

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:

YAML

```
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
```

JSON

```
{
  "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"
 }
```

In alternativa, puoi creare un file backend utilizzando le credenziali SVM (nome utente e password) memorizzate in AWS Secret Manager, come mostrato in questo esempio:

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  managementLIF:
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFilesystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-
2:xxxxxxxx:secret:secret-name",
      "type": "awsarn"
    }
  }
}
```

Per informazioni sulla creazione di backend, consulta i seguenti <a href="https://docs.netapp.com/it-it/trident-2402/trident-use/+
  xref:./trident-use/ontap-nas.html">"Configurare un backend con i driver NAS ONTAP"
  "Configurare un backend con i driver SAN ONTAP"

Preparatevi al provisioning dei volumi SMB

È possibile eseguire il provisioning dei volumi SMB utilizzando `ontap-nas` driver. Prima di completare Integrazione dei driver ONTAP SAN e NAS completare i seguenti passaggi.

Prima di iniziare

Prima di eseguire il provisioning di volumi SMB utilizzando `ontap-nas` driver, è necessario disporre di quanto segue.

- Un cluster Kubernetes con un nodo controller Linux e almeno un nodo di lavoro Windows che esegue Windows Server 2019. Astra Trident supporta volumi SMB montati su pod eseguiti solo su nodi Windows.
- Almeno un segreto Astra Trident contenente le credenziali Active Directory. Per generare un segreto `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Proxy CSI configurato come servizio Windows. Per configurare un `csi-proxy`, fare riferimento a. "GitHub: Proxy CSI" oppure "GitHub: Proxy CSI per Windows" Per i nodi Kubernetes in esecuzione su Windows.

Fasi

1. Creare condivisioni SMB. È possibile creare le condivisioni amministrative SMB in due modi utilizzando "Console di gestione Microsoft" Snap-in cartelle condivise o utilizzo dell'interfaccia CLI di ONTAP. Per creare le condivisioni SMB utilizzando la CLI ONTAP:
   a. Se necessario, creare la struttura del percorso di directory per la condivisione.

      Il `vserver cifs share create` il comando controlla il percorso specificato nell'opzione -path durante la creazione della condivisione. Se il percorso specificato non esiste, il comando non riesce.

   b. Creare una condivisione SMB associata alla SVM specificata:

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

   c. Verificare che la condivisione sia stata creata:

```
vserver cifs share show -share-name share_name
```

      Fare riferimento a. "Creare una condivisione SMB" per informazioni dettagliate.

2. Quando si crea il backend, è necessario configurare quanto segue per specificare i volumi SMB. Per tutte le opzioni di configurazione backend FSX per ONTAP, fare riferimento a. "FSX per le opzioni di configurazione e gli esempi di ONTAP".

Parametro	Descrizione	Esempio
`smbShare`	È possibile specificare una delle seguenti opzioni: Il nome di una condivisione SMB creata utilizzando la console di gestione Microsoft o l'interfaccia utente di ONTAP o un nome per consentire ad Astra Trident di creare la condivisione SMB. Questo parametro è obbligatorio per i backend Amazon FSX per ONTAP.	`smb-share`
`nasType`	**Deve essere impostato su `smb`.** se null, il valore predefinito è `nfs`.	`smb`
`securityStyle`	Stile di sicurezza per nuovi volumi. **Deve essere impostato su `ntfs` oppure `mixed` Per volumi SMB.**	`ntfs` oppure `mixed` Per volumi SMB
`unixPermissions`	Per i nuovi volumi. **Deve essere lasciato vuoto per i volumi SMB.**	""

FSX per le opzioni di configurazione e gli esempi di ONTAP

Scopri le opzioni di configurazione back-end per Amazon FSX per ONTAP. Questa sezione fornisce esempi di configurazione back-end.

Opzioni di configurazione back-end

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

Parametro	Descrizione	Esempio
`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

Parametro	Descrizione	Esempio
`managementLIF`	Indirizzo IP di un cluster o LIF di gestione SVM È possibile specificare un nome di dominio completo (FQDN). Può essere impostato per utilizzare gli indirizzi IPv6 se Astra Trident è stato installato utilizzando il flag IPv6. Gli indirizzi IPv6 devono essere definiti tra parentesi quadre, ad esempio [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Indirizzo IP del protocollo LIF. **Driver NAS ONTAP**: Si consiglia di specificare dataLIF. Se non fornito, Astra Trident recupera i dati LIF dalla SVM. È possibile specificare un FQDN (Fully-qualified domain name) da utilizzare per le operazioni di montaggio NFS, consentendo di creare un DNS round-robin per il bilanciamento del carico tra più LIF di dati. Può essere modificato dopo l'impostazione iniziale. Fare riferimento a. . **Driver SAN ONTAP**: Non specificare iSCSI. Astra Trident utilizza la mappa LUN selettiva di ONTAP per rilevare le LIF iSCI necessarie per stabilire una sessione multi-percorso. Viene generato un avviso se dataLIF è esplicitamente definito. Può essere impostato per utilizzare gli indirizzi IPv6 se Astra Trident è stato installato utilizzando il flag IPv6. Gli indirizzi IPv6 devono essere definiti tra parentesi quadre, ad esempio [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	

Parametro	Descrizione	Esempio
`autoExportPolicy`	Abilita la creazione e l'aggiornamento automatici dei criteri di esportazione [booleano]. Utilizzando il `autoExportPolicy` e. `autoExportCIDRs` Astra Trident è in grado di gestire automaticamente le policy di esportazione.	`false`
`autoExportCIDRs`	Elenco di CIDR per filtrare gli IP dei nodi di Kubernetes rispetto a quando `autoExportPolicy` è attivato. Utilizzando il `autoExportPolicy` e. `autoExportCIDRs` Astra Trident è in grado di gestire automaticamente le policy di esportazione.	"["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	""
`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 o alla SVM. Utilizzato per l'autenticazione basata su credenziali. Ad esempio, vsadmin.	
`password`	Password per la connessione al cluster o alla SVM. Utilizzato per l'autenticazione basata su credenziali.	
`svm`	Macchina virtuale per lo storage da utilizzare	Derivato se viene specificato un LIF di gestione SVM.

Parametro	Descrizione	Esempio
`storagePrefix`	Prefisso utilizzato per il provisioning di nuovi volumi nella SVM. Impossibile modificare dopo la creazione. Per aggiornare questo parametro, è necessario creare un nuovo backend.	`trident`
`limitAggregateUsage`	**Non specificare Amazon FSX per NetApp ONTAP.** Il fornito `fsxadmin` e. `vsadmin` Non includere le autorizzazioni necessarie per recuperare l'utilizzo aggregato e limitarlo utilizzando Astra Trident.	Non utilizzare.
`limitVolumeSize`	Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore. Inoltre, limita le dimensioni massime dei volumi gestiti per qtree e LUN, oltre a `qtreesPerFlexvol` Consente di personalizzare il numero massimo di qtree per FlexVol.	"" (non applicato per impostazione predefinita)
`lunsPerFlexvol`	Il numero massimo di LUN per FlexVol deve essere compreso nell'intervallo [50, 200]. Solo SAN.	`100`
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Ad esempio, {"api":false,} method":true Non utilizzare `debugTraceFlags` a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.	nullo

Parametro	Descrizione	Esempio
`nfsMountOptions`	Elenco separato da virgole delle opzioni di montaggio NFS. 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.	""
`nasType`	Configurare la creazione di volumi NFS o SMB. Le opzioni sono `nfs`, `smb`o `nullo`. **Deve essere impostato su `smb`** `Per i volumi SMB.` l'impostazione su Null imposta come predefinita i volumi NFS.	`nfs`
`qtreesPerFlexvol`	Qtree massimi per FlexVol, devono essere compresi nell'intervallo [50, 300]	`200`
`smbShare`	È possibile specificare una delle seguenti opzioni: Il nome di una condivisione SMB creata utilizzando la console di gestione Microsoft o l'interfaccia utente di ONTAP o un nome per consentire ad Astra Trident di creare la condivisione SMB. Questo parametro è obbligatorio per i backend Amazon FSX per ONTAP.	`smb-share`

Parametro	Descrizione	Esempio
`useREST`	Parametro booleano per l'utilizzo delle API REST di ONTAP. **Anteprima tecnica** `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 9.11.1 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.	`false`
`aws`	Puoi specificare quanto segue nel file di configurazione per AWS FSX per ONTAP: - `fsxFilesystemID`: Specificare l'ID del file system AWS FSX. - `apiRegion`: Nome regione API AWS. - `apikey`: Chiave API AWS. - `secretKey`: Chiave segreta AWS.	"" "" ""
`credentials`	Specifica le credenziali della SVM di FSX da archiviare in AWS Secret Manager. - `name`: Amazon Resource Name (ARN) del segreto, che contiene le credenziali di SVM. - `type`: Impostare su `awsarn`. Fare riferimento a. "Creare un segreto AWS Secrets Manager" per ulteriori informazioni.	

Aggiornare `dataLIF` dopo la configurazione iniziale

È possibile modificare la LIF dei dati dopo la configurazione iniziale eseguendo il seguente comando per fornire al nuovo file JSON di back-end i dati aggiornati LIF.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

Se i PVC sono collegati a uno o più pod, è necessario abbassare tutti i pod corrispondenti e riportarli di nuovo in alto per rendere effettiva la nuova LIF dei dati.

Opzioni di configurazione back-end per il provisioning dei volumi

È possibile controllare il provisioning predefinito utilizzando queste opzioni in `defaults` della configurazione. Per un esempio, vedere gli esempi di configurazione riportati di seguito.

Parametro	Descrizione	Predefinito
`spaceAllocation`	Allocazione dello spazio per LUN	`true`
`spaceReserve`	Modalità di riserva dello spazio; "nessuno" (sottile) o "volume" (spesso)	`none`
`snapshotPolicy`	Policy di Snapshot da utilizzare	`none`
`qosPolicy`	Gruppo di criteri QoS da assegnare per i volumi creati. Scegliere una delle opzioni qosPolicy o adaptiveQosPolicy per pool di storage o backend. L'utilizzo di gruppi di policy QoS con Astra Trident richiede ONTAP 9.8 o versione successiva. Si consiglia di utilizzare un gruppo di policy QoS non condiviso e di assicurarsi che il gruppo di policy venga 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.	""
`adaptiveQosPolicy`	Gruppo di criteri QoS adattivi da assegnare per i volumi creati. Scegliere una delle opzioni qosPolicy o adaptiveQosPolicy per pool di storage o backend. Non supportato da ontap-nas-Economy.	""
`snapshotReserve`	Percentuale di volume riservato agli snapshot "0"	Se `snapshotPolicy` è `none`, `else` ""
`splitOnClone`	Separare un clone dal suo padre al momento della creazione	`false`

Parametro	Descrizione	Predefinito
`encryption`	Abilitare NetApp Volume Encryption (NVE) sul nuovo volume; il valore predefinito è `false`. NVE deve essere concesso in licenza e abilitato sul cluster per utilizzare questa opzione. Se NAE è attivato sul backend, tutti i volumi forniti in Astra Trident saranno abilitati per NAE. Per ulteriori informazioni, fare riferimento a: "Come funziona Astra Trident con NVE e NAE".	`false`
`luksEncryption`	Attivare la crittografia LUKS. Fare riferimento a. "Utilizzo di Linux Unified Key Setup (LUKS)". Solo SAN.	""
`tieringPolicy`	Policy di tiering da utilizzare `none`	`snapshot-only` Per la configurazione SVM-DR precedente a ONTAP 9.5
`unixPermissions`	Per i nuovi volumi. **Lasciare vuoto per i volumi SMB.**	""
`securityStyle`	Stile di sicurezza per nuovi volumi. Supporto di NFS `mixed` e. `unix` stili di sicurezza. Supporto SMB `mixed` e. `ntfs` stili di sicurezza.	Il valore predefinito di NFS è `unix`. Il valore predefinito di SMB è `ntfs`.

Configurazioni di esempio

Configurazione della classe di storage per volumi SMB

Utilizzo di `nasType`, `node-stage-secret-name`, e. `node-stage-secret-namespace`, È possibile specificare un volume SMB e fornire le credenziali Active Directory richieste. I volumi SMB sono supportati utilizzando `ontap-nas` solo driver.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: nas-smb-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

Configurazione per AWS FSX per ONTAP con gestore segreto

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  managementLIF:
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

Configurare il componente aggiuntivo Astra Trident EKS versione 23,10 sul cluster EKS

Astra Trident ottimizza la gestione dello storage di Amazon FSX per NetApp ONTAP in Kubernetes per permettere a sviluppatori e amministratori di concentrarsi sull'implementazione dell'applicazione. Il componente aggiuntivo Astra Trident EKS include le più recenti patch di sicurezza, correzioni di bug ed è convalidato da AWS per funzionare con Amazon EKS. Il componente aggiuntivo EKS ti consente di garantire in

modo coerente che i tuoi cluster Amazon EKS siano sicuri e stabili e di ridurre la quantità di lavoro da svolgere per installare, configurare e aggiornare i componenti aggiuntivi.

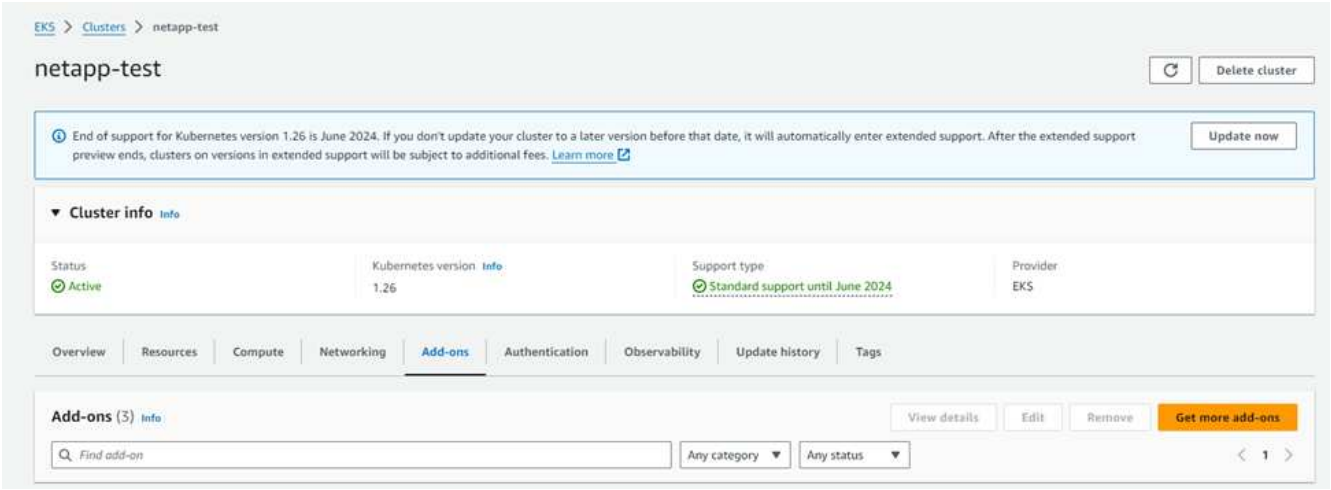
Prerequisiti

Prima di configurare il componente aggiuntivo Astra Trident per AWS EKS, assicurati di disporre di quanto segue:

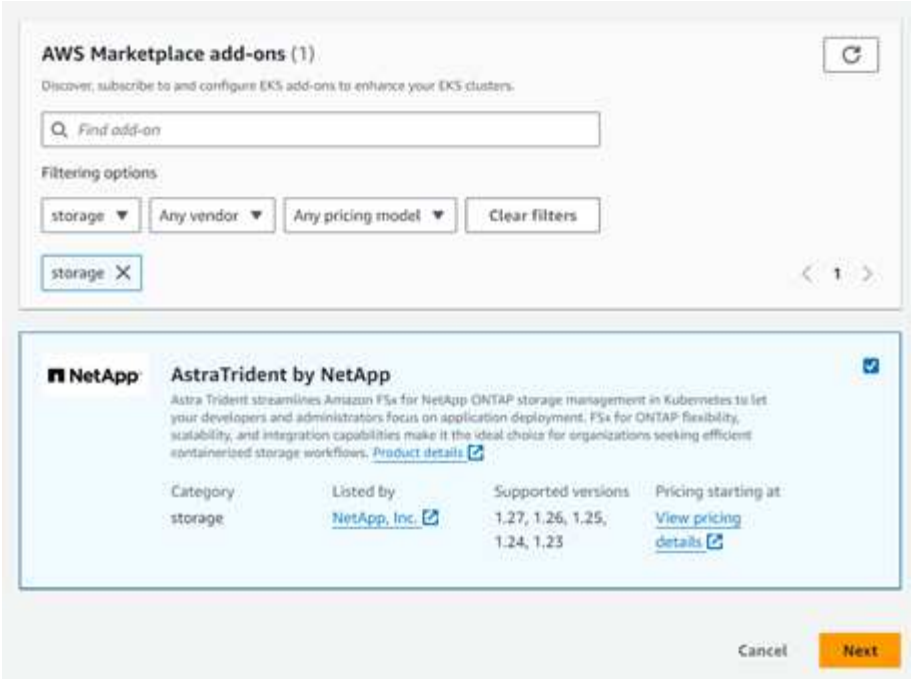
- Un account cluster Amazon EKS con abbonamento add-on
- Autorizzazioni AWS nel marketplace AWS:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Tipo di ami: Amazon Linux 2 (AL2_x86_64) o Amazon Linux 2 Arm (AL2_ARM_64)
- Tipo di nodo: AMD o ARM
- Un file system Amazon FSX per NetApp ONTAP esistente

Fasi

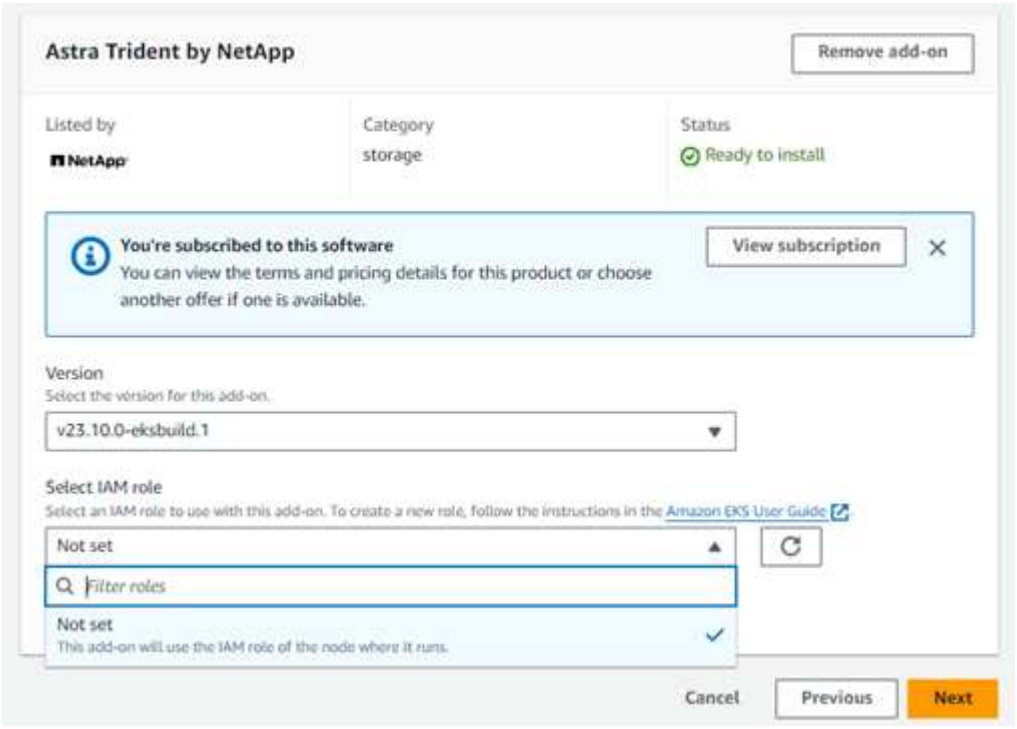
1. Sul tuo cluster EKS Kubernetes, accedi alla scheda **Add-on**.



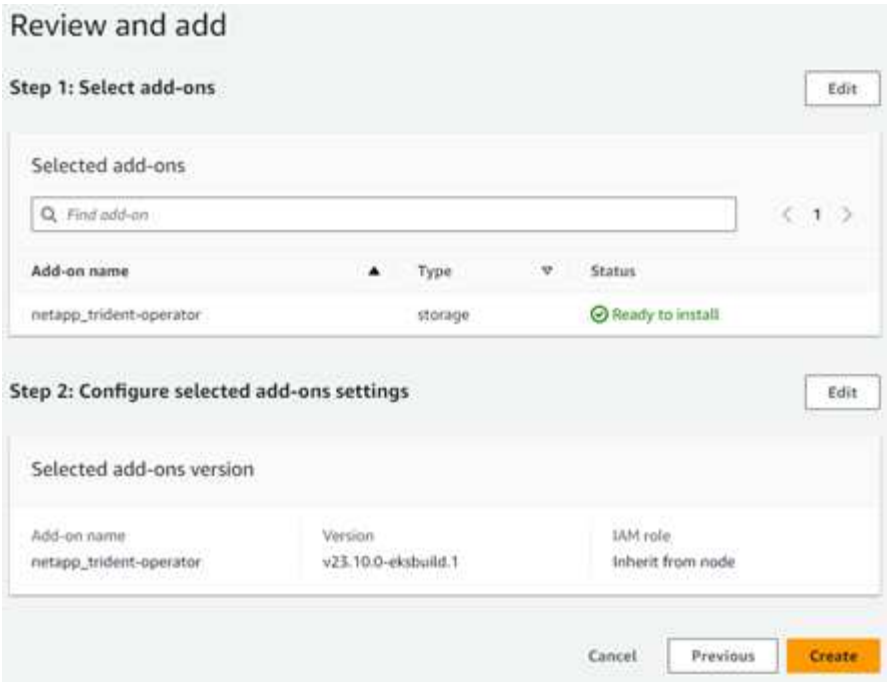
2. Vai su **componenti aggiuntivi di AWS Marketplace** e scegli la categoria *storage*.



3. Individua **AstraTrident by NetApp** e seleziona la casella di controllo per il componente aggiuntivo Astra Trident.
4. Scegliere la versione desiderata del componente aggiuntivo.



5. Selezionare l'opzione ruolo IAM per ereditare dal nodo.
6. Configurare eventuali impostazioni opzionali secondo necessità e selezionare **Avanti**.



7. Selezionare **Crea**.
8. Verificare che lo stato del componente aggiuntivo sia *attivo*.

Installare/disinstallare il componente aggiuntivo Astra Trident EKS utilizzando la CLI

Installare il componente aggiuntivo Astra Trident EKS utilizzando la CLI:

I seguenti comandi di esempio installano il componente aggiuntivo Astra Trident EKS:
`eksctl create addon --cluster K8s-arm --name netapp_trident-operator --version v23.10.0-eksbuild.`
`eksctl create addon --cluster K8s-arm --name netapp_trident-operator --version v23.10.0-eksbuild.1` (con una versione dedicata)

Disinstallare il componente aggiuntivo Astra Trident EKS utilizzando la CLI:

Il seguente comando disinstalla il componente aggiuntivo Astra Trident EKS:
`eksctl delete addon --cluster K8s-arm --name netapp_trident-operator`

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 un elenco delle opzioni di configurazione del driver di archiviazione desiderato, consultare la "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: password
```

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
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

Descrizione dei campi segreti della piattaforma di storage	Segreto	Descrizione dei campi
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` 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".

Gestire i backend

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).
  - È possibile eseguire l'aggiornamento `TridentBackendConfig` Oggetti direttamente tramite Kubernetes usando il seguente comando:

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

  - In alternativa, è possibile apportare modifiche all'esistente `TridentBackendConfig` CR utilizzando 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.

Opzioni per la gestione dei backend

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 hanno un `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 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 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.

Creare e gestire classi di archiviazione

Creare una classe di storage

Configurare un oggetto Kubernetes StorageClass e creare una classe storage per istruire Astra Trident su come eseguire il provisioning dei volumi.

Configurare un oggetto Kubernetes StorageClass

Il "Oggetto Kubernetes StorageClass" Identifica Astra Trident come provisioner utilizzato per quella classe e istruisce Astra Trident su come eseguire il provisioning di un volume. Ad esempio:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters:
  <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Fare riferimento a. "Kubernetes e Trident Objects" per informazioni sulle modalità di interazione delle classi di storage con `PersistentVolumeClaim` E parametri per controllare come Astra Trident esegue il provisioning dei volumi.

Creare una classe di storage

Dopo aver creato l'oggetto StorageClass, è possibile creare la classe storage. Campioni di classe di conservazione fornisce alcuni esempi di base che è possibile utilizzare o modificare.

Fasi

1. Si tratta di un oggetto Kubernetes, lo utilizza `kubectl` Per crearlo in Kubernetes.

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

2. 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"
        ]
      }
    }
  ]
}
```

Campioni di classe di conservazione

Astra Trident offre "definizioni semplici delle classi di archiviazione per backend specifici".

In alternativa, è possibile modificare `sample-input/storage-class-csi.yaml.templ` file fornito con il programma di installazione e sostituirlo *`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
```

Gestire le classi di storage

È possibile visualizzare le classi di storage esistenti, impostare una classe di storage predefinita, identificare il backend della classe di storage ed eliminare le classi di storage.

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 deve essere presente una sola classe di archiviazione predefinita alla 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}]'
```

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 le StorageClasses esistenti e ricrearle con `parameters.fsType` specificato.

Provisioning e gestione dei volumi

Provisioning di un volume

Creare un PersistentVolume (PV) e un PersistentVolumeClaim (PVC) che utilizza Kubernetes StorageClass configurato per richiedere l'accesso al PV. È quindi possibile montare il PV su un pod.

Panoramica

R "*PersistentVolume*" (PV) è una risorsa di storage fisico fornita dall'amministratore del cluster su un cluster Kubernetes. Il "*PersistentVolumeClaim*" (PVC) è una richiesta di accesso a PersistentVolume sul cluster.

Il PVC può essere configurato per richiedere la memorizzazione di una determinata dimensione o modalità di accesso. Utilizzando StorageClass associato, l'amministratore del cluster può controllare più delle dimensioni di PersistentVolume e della modalità di accesso, ad esempio le prestazioni o il livello di servizio.

Dopo aver creato PV e PVC, è possibile montare il volume in un pod.

Manifesti campione

Manifesto di esempio di PersistentVolume

Questo manifesto di esempio mostra un PV di base di 10Gi associato a StorageClass `basic-csi`.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-storage
  labels:
    type: local
spec:
  storageClassName: basic-csi
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/my/host/path"
```

Manifesti di campioni PersistentVolumeClaim

Questi esempi mostrano le opzioni di configurazione di base del PVC.

PVC con accesso RWO

Questo esempio mostra un PVC di base con accesso RWO associato a un nome StorageClass `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

PVC con NVMe/TCP

Questo esempio mostra un PVC di base per NVMe/TCP con accesso RWO associato a una StorageClass denominata `protection-gold`.

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san-nvme
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: protection-gold
```

Campioni manifesti pod

Questi esempi mostrano le configurazioni di base per collegare il PVC a un pod.

Configurazione di base

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

Configurazione NVMe/TCP di base

```
---
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
    - image: nginx
      name: nginx
      resources: {}
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
      claimName: pvc-san-nvme
```

Creare PV e PVC

Fasi

1. Creare il PV.

```
kubectl create -f pv.yaml
```

2. Verificare lo stato FV.

```
kubectl get pv
NAME        CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS    CLAIM
STORAGECLASS  REASON  AGE
pv-storage  4Gi       RWO           Retain          Available
7s
```

3. Creare il PVC.

```
kubectl create -f pvc.yaml
```

4. Verificare lo stato del PVC.

```
kubectl get pvc
NAME        STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage Bound  pv-name 2Gi      RWO                          5m
```

5. Montare il volume in un pod.

```
kubectl create -f pv-pod.yaml
```

> È possibile monitorare l'avanzamento utilizzando `kubectl get pod --watch`.

6. Verificare che il volume sia montato su `/my/mount/path`.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

7. A questo punto è possibile eliminare il pod. L'applicazione Pod non esisterà più, ma il volume rimarrà.

```
kubectl delete pod task-pv-pod
```

Fare riferimento a. "Kubernetes e Trident Objects" per informazioni sulle modalità di interazione delle classi di storage con `PersistentVolumeClaim` E parametri per controllare come Astra Trident esegue il provisioning dei volumi.

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.

Fase 1: Configurare StorageClass per supportare l'espansione dei volumi

Modificare la definizione StorageClass per impostare `allowVolumeExpansion` campo a. `true`.

```
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

Modificare la definizione PVC e aggiornare `spec.resources.requests.storage` per riflettere le nuove dimensioni desiderate, che devono essere superiori alle dimensioni originali.

```
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

Collegare il PV a un pod affinché venga 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.

In questo esempio, viene creato un pod che utilizza `san-pvc`.

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-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:    ubuntu-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 1GiB:

```
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`.

Panoramica e considerazioni

È possibile importare un volume in Astra Trident per:

- Containerizzare un'applicazione e riutilizzare il set di dati esistente
- Utilizzare un clone di un set di dati per un'applicazione temporanea
- Ricostruire un cluster Kubernetes guasto
- Migrazione dei dati delle applicazioni durante il disaster recovery

Considerazioni

Prima di importare un volume, esaminare le seguenti considerazioni.

- Astra Trident può importare solo volumi ONTAP di tipo RW (Read-write). I volumi di tipo DP (data Protection) sono volumi di destinazione SnapMirror. Prima di importare il volume in Astra Trident, è necessario interrompere la relazione di mirroring.

- Si consiglia di importare volumi senza connessioni attive. Per importare un volume utilizzato attivamente, clonare il volume ed eseguire l'importazione.

  Ciò è particolarmente importante per i volumi a blocchi, in quanto Kubernetes non sarebbe a conoscenza della connessione precedente e potrebbe facilmente collegare un volume attivo a un pod. Ciò può causare il danneggiamento dei dati.

- Tuttavia `StorageClass` Deve essere specificato su PVC, Astra Trident non utilizza questo parametro durante l'importazione. Le classi di storage vengono utilizzate durante la creazione del volume per selezionare i pool disponibili in base alle caratteristiche dello storage. Poiché il volume esiste già, durante l'importazione non è richiesta alcuna selezione del pool. Pertanto, l'importazione non avrà esito negativo anche se il volume esiste in un backend o in un pool che non corrisponde alla classe di storage specificata nel PVC.
- La dimensione del volume esistente viene determinata e impostata 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.
- Per impostazione predefinita, Astra Trident gestisce il PVC e rinomina il FlexVol e il LUN sul backend. È possibile superare il `--no-manage` contrassegna per importare un volume non gestito. Se si utilizza `--no-manage`, Astra Trident non esegue operazioni aggiuntive sul PVC o sul PV per il ciclo di vita degli oggetti. Il volume di storage non viene cancellato quando il PV viene cancellato e 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.

Importare un volume

È possibile utilizzare `tridentctl import` per importare un volume.

Fasi

1. Creare il file PVC (Persistent Volume Claim) (ad esempio, `pvc.yaml`) Che verrà utilizzato per creare il PVC. Il file PVC deve includere `name`, `namespace`, `accessModes`, e. `storageClassName`. In alternativa, è possibile specificare `unixPermissions` Nella definizione di PVC.

   Di seguito viene riportato un esempio di specifica minima:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```

Non includere parametri aggiuntivi come il nome PV o le dimensioni del volume. Questo può causare l'errore del comando di importazione.

2. Utilizzare `tridentctl import` Per specificare il nome del backend Astra Trident contenente il volume e il nome che identifica in modo univoco il volume nello storage (ad esempio: ONTAP FlexVol, volume elemento, percorso Cloud Volumes Service). Il `-f` Argomento necessario per specificare il percorso del file PVC.

```
tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-file>
```

Esempi

Consultare i seguenti esempi di importazione di volumi per i driver supportati.

NAS ONTAP e NAS FlexGroup ONTAP

Astra Trident supporta l'importazione di volumi utilizzando `ontap-nas` e. `ontap-nas-flexgroup` driver.

- Il `ontap-nas-economy` il driver non può importare e gestire qtree.
- Il `ontap-nas` e. `ontap-nas-flexgroup` i driver non consentono nomi di volumi duplicati.

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.

Esempi di NAS ONTAP

Di seguito viene illustrato un esempio di importazione di un volume gestito e di un volume non gestito.

Volume gestito

Nell'esempio seguente viene importato un volume denominato `managed_volume` su un backend denominato `ontap_nas`:

```
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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Volume non gestito

Quando si utilizza `--no-manage` Argomento: Astra Trident non rinomina il volume.

L'esempio seguente importa `unmanaged_volume` su `ontap_nas` back-end:

```
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   |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

ONTAP SAN

Astra Trident supporta l'importazione di volumi utilizzando `ontap-san` driver. L'importazione di un volume non è supportata con `ontap-san-economy` driver.

Astra Trident può 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. Astra Trident importa il FlexVol e lo associa alla definizione del PVC.

Esempi DI SAN ONTAP

Di seguito viene illustrato un esempio di importazione di un volume gestito e di un volume non gestito.

Volume gestito

Per i volumi gestiti, Astra Trident rinomina FlexVol in `pvc-<uuid>` E il LUN all'interno di FlexVol a. `lun0`.

Nell'esempio riportato di seguito viene importato il `ontap-san-managed` FlexVol presente su `ontap_san_default` back-end:

```
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    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

Volume non gestito

L'esempio seguente importa `unmanaged_example_volume` su `ontap_san` back-end:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

Se si dispone DI LUN mappati a igroups che condividono un IQN con un nodo Kubernetes IQN, come mostrato nell'esempio seguente, viene visualizzato l'errore: `LUN already mapped to initiator(s) in this group`. Per importare il volume, è necessario rimuovere l'iniziatore o annullare la mappatura del LUN.

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

Elemento

Astra Trident supporta il software NetApp Element e l'importazione di volumi NetApp HCI utilizzando `solidfire-san` driver.

> Il driver Element supporta nomi di volumi duplicati. Tuttavia, Astra Trident restituisce un errore se sono presenti nomi di volumi duplicati. Come soluzione alternativa, clonare il volume, fornire un nome di volume univoco e importare il volume clonato.

Esempio di elemento

Nell'esempio seguente viene importato un `element-managed` volume sul back-end `element_default`.

```
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    |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
```

Piattaforma Google Cloud

Astra Trident supporta l'importazione di volumi utilizzando `gcp-cvs` driver.

> Per importare un volume supportato da NetApp Cloud Volumes Service in Google Cloud Platform, identificare il volume in base al relativo percorso. 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`.

Esempio di piattaforma Google Cloud

Nell'esempio seguente viene importato un `gcp-cvs` volume sul back-end `gcpcvs_YEppr` con il percorso del volume di `adroit-jolly-swift`.

```
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    |
+------------------------------------------+--------+--------------
+----------+------------------------------------+--------+---------+
```

Azure NetApp Files

Astra Trident supporta l'importazione di volumi utilizzando `azure-netapp-files` driver.

> Per importare un volume Azure NetApp Files, identificare il volume in base al relativo percorso. Il percorso del volume è la parte del percorso di esportazione del volume dopo `:/`. Ad esempio, se il percorso di montaggio è `10.0.0.2:/importvol1`, il percorso del volume è `importvol1`.

Esempio di Azure NetApp Files

Nell'esempio seguente viene importato un `azure-netapp-files` volume sul back-end `azurenetappfiles_40517` con il percorso del volume `importvol1`.

```
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    |
+------------------------------------------+---------+--------------
+----------+------------------------------------+--------+---------+
```

Condividere un volume NFS tra spazi dei nomi

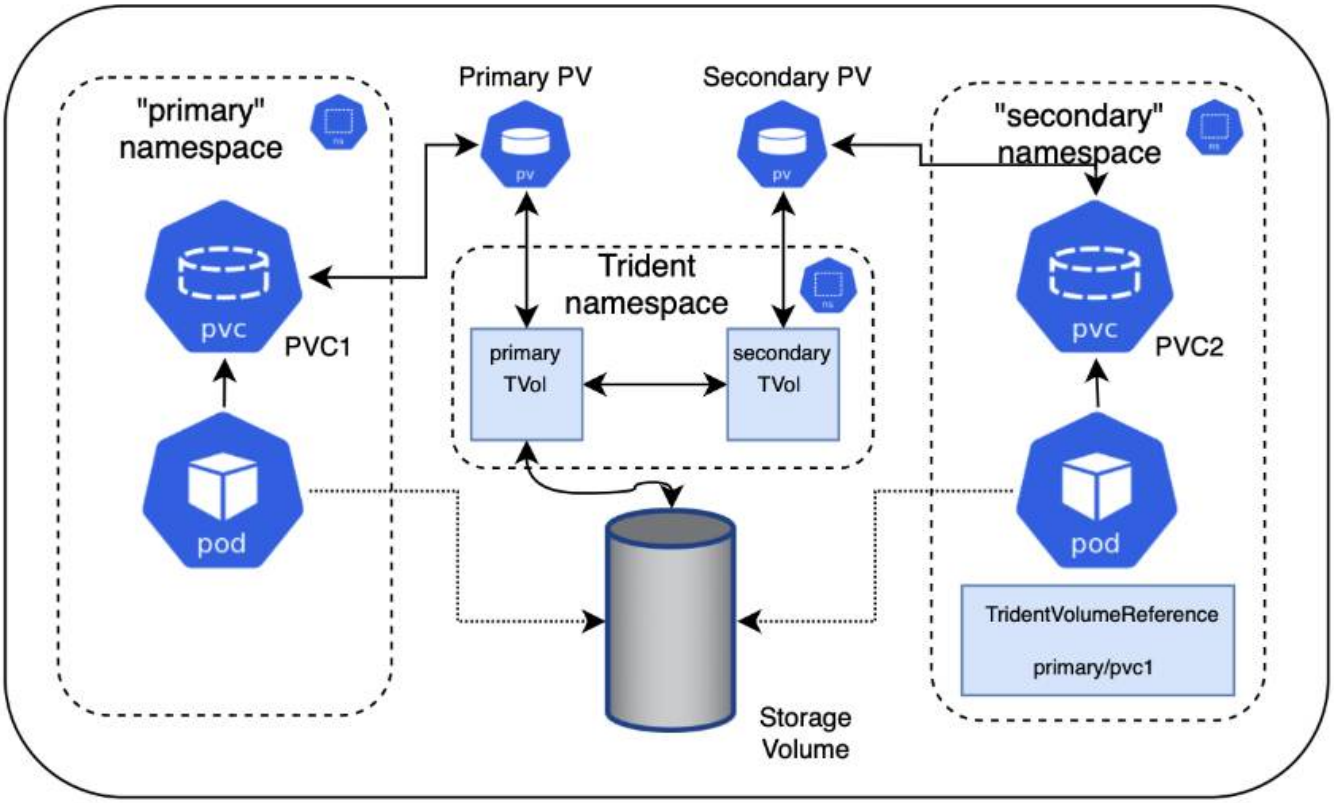
Utilizzando Astra Trident, è possibile creare un volume in uno spazio dei nomi primario e condividerlo in uno o più spazi dei nomi secondari.

Caratteristiche

Astra TridentVolumeReference CR consente di condividere in modo sicuro volumi NFS ReadWriteMany (RWX) in uno o più spazi dei nomi Kubernetes. Questa soluzione nativa di Kubernetes offre i seguenti vantaggi:

- Diversi livelli di controllo degli accessi per garantire la sicurezza
- Funziona con tutti i driver di volume NFS Trident
- Nessuna dipendenza da tridentctl o da altre funzionalità Kubernetes non native

Questo diagramma illustra la condivisione del volume NFS tra due spazi dei nomi Kubernetes.



Avvio rapido

Puoi configurare la condivisione dei volumi NFS in pochi passaggi.

1 **Configurare il PVC di origine per la condivisione del volume**

Il proprietario dello spazio dei nomi di origine concede il permesso di accedere ai dati nel PVC di origine.

2 **Concedere il permesso di creare una CR nello spazio dei nomi di destinazione**

L'amministratore del cluster concede l'autorizzazione al proprietario dello spazio dei nomi di destinazione per creare la CR di TridentVolumeReference.

3 Creare TridentVolumeReference nello spazio dei nomi di destinazione

Il proprietario dello spazio dei nomi di destinazione crea la CR di TridentVolumeReference per fare riferimento al PVC di origine.

4 Creare il PVC subordinato nello spazio dei nomi di destinazione

Il proprietario dello spazio dei nomi di destinazione crea il PVC subordinato per utilizzare l'origine dati dal PVC di origine.

Configurare gli spazi dei nomi di origine e di destinazione

Per garantire la sicurezza, la condivisione di spazi dei nomi incrociati richiede la collaborazione e l'azione del proprietario dello spazio dei nomi di origine, dell'amministratore del cluster e del proprietario dello spazio dei nomi di destinazione. Il ruolo dell'utente viene designato in ogni fase.

Fasi

1. **Source namespace owner:** Crea il PVC (`pvc1`) nello spazio dei nomi di origine che concede l'autorizzazione per la condivisione con lo spazio dei nomi di destinazione (`namespace2`) utilizzando `shareToNamespace` annotazione.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

   Astra Trident crea il PV e il suo volume di storage NFS back-end.

- È possibile condividere il PVC con più spazi dei nomi utilizzando un elenco delimitato da virgole. Ad esempio, `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4`.
- È possibile condividere con tutti gli spazi dei nomi utilizzando `*`. Ad esempio, `trident.netapp.io/shareToNamespace: *`
- È possibile aggiornare il PVC per includere `shareToNamespace` annotazione in qualsiasi momento.

2. **Cluster admin:** creare il ruolo personalizzato e il kubeconfig per concedere l'autorizzazione al proprietario dello spazio dei nomi di destinazione per creare il CR di TridentVolumeReference nello spazio dei nomi di destinazione.
3. **Destination namespace owner:** creare una CR di TridentVolumeReference nello spazio dei nomi di destinazione che si riferisce allo spazio dei nomi di origine `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Proprietario dello spazio dei nomi di destinazione:** Crea un PVC (`pvc2`) nello spazio dei nomi di destinazione (`namespace2`) utilizzando `shareFromPVC` Annotazione per indicare il PVC di origine.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

La dimensione del PVC di destinazione deve essere inferiore o uguale al PVC di origine.

Risultati

Astra Trident legge `shareFromPVC` Annotazione sul PVC di destinazione e crea il PV di destinazione come volume subordinato senza una propria risorsa di storage che punta al PV di origine e condivide la risorsa di storage PV di origine. Il PVC e il PV di destinazione appaiono associati come normali.

Eliminare un volume condiviso

È possibile eliminare un volume condiviso tra più spazi dei nomi. Astra Trident rimuoverà l'accesso al volume nello spazio dei nomi di origine e manterrà l'accesso ad altri spazi dei nomi che condividono il volume. Una volta rimossi tutti gli spazi dei nomi che fanno riferimento al volume, Astra Trident elimina il volume.

Utilizzare `tridentctl get` per eseguire query sui volumi subordinati

Utilizzando il[`tridentctl` è possibile eseguire `get` comando per ottenere volumi subordinati. Per ulteriori informazioni, fare riferimento al `tridentctl` comandi e opzioni.

```
Usage:
  tridentctl get [option]
```

Allarmi:

- `-h, --help`: Guida per i volumi.
- `--parentOfSubordinate string`: Limita query al volume di origine subordinato.
- `--subordinateOf string`: Limita la query alle subordinate del volume.

Limitazioni

- Astra Trident non può impedire la scrittura degli spazi dei nomi di destinazione nel volume condiviso. È necessario utilizzare il blocco dei file o altri processi per impedire la sovrascrittura dei dati dei volumi condivisi.
- Non è possibile revocare l'accesso al PVC di origine rimuovendo `shareToNamespace` oppure `shareFromNamespace` annotazioni o eliminazione di `TridentVolumeReference` CR. Per revocare l'accesso, è necessario eliminare il PVC subordinato.
- Snapshot, cloni e mirroring non sono possibili sui volumi subordinati.

Per ulteriori informazioni

Per ulteriori informazioni sull'accesso ai volumi tra spazi dei nomi:

- Visitare il sito "Condivisione di volumi tra spazi dei nomi: Dai il benvenuto all'accesso a volumi tra spazi dei nomi".

Replica dei volumi con SnapMirror

Con Astra Control Provivisioner, puoi creare relazioni di mirroring tra un volume di origine su un cluster e il volume di destinazione sul cluster in peering per replicare i dati per il disaster recovery. È possibile utilizzare una definizione di risorsa personalizzata (CRD) con nome per eseguire le seguenti operazioni:

- Creare relazioni di mirroring tra volumi (PVC)
- Rimuovere le relazioni di mirroring tra volumi
- Interrompere le relazioni di mirroring
- Promozione del volume secondario in condizioni di disastro (failover)
- Eseguire la transizione senza perdita di dati delle applicazioni da cluster a cluster (durante failover o migrazioni pianificati)

Prerequisiti per la replica

Prima di iniziare, verificare che siano soddisfatti i seguenti prerequisiti:

Cluster ONTAP

- **Astra Control Provisioner**: Astra Control Provisioner versione 23,10 o successiva deve esistere sia sui cluster Kubernetes di origine che di destinazione che utilizzano ONTAP come backend.
- **Licenze**: Le licenze asincrone di ONTAP SnapMirror che utilizzano il bundle di protezione dati devono essere attivate sia sul cluster ONTAP di origine che su quello di destinazione. Per ulteriori informazioni, fare riferimento "Panoramica sulle licenze SnapMirror in ONTAP" a.

Peering

- **Cluster e SVM**: I backend dello storage ONTAP devono essere peering. Per ulteriori informazioni, fare riferimento "Panoramica del peering di cluster e SVM" a.

  > Assicurati che i nomi delle SVM utilizzati nella relazione di replica tra due cluster ONTAP siano univoci.

- **Astra Control Provisioner e SVM**: Le SVM remote in cui è stato eseguito il peering devono essere disponibili per Astra Control Provisioner nel cluster di destinazione.

Driver supportati

- La replica di un volume è supportata per i driver ontap-nas e ontap-san.

Creare un PVC specchiato

Seguire questi passaggi e utilizzare gli esempi CRD per creare una relazione di mirroring tra volumi primari e secondari.

Fasi

1. Eseguire i seguenti passaggi sul cluster Kubernetes primario:
   a. Creare un oggetto StorageClass con il `trident.netapp.io/replication: true` parametro.

Esempio

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "nfs"
  trident.netapp.io/replication: "true"
```

b. Crea un PVC con StorageClass creato in precedenza.

Esempio

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

c. Creare una CR MirrorRelationship con informazioni locali.

Esempio

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Astra Control Provivisioner recupera le informazioni interne del volume e dello stato attuale di data Protection (DP) del volume, quindi popola il campo di stato di MirrorRelationship.

d. Procurarsi il TridentMirrorRelationship CR per ottenere il nome interno e la SVM del PVC.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. Eseguire i seguenti passaggi sul cluster Kubernetes secondario:
   a. Creare una classe StorageClass con il parametro trident.netapp.io/replication: true.

      Esempio

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        trident.netapp.io/replication: true
      ```

   b. Creare una CR MirrorRelationship con informazioni sulla destinazione e sulla sorgente.

Esempio

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Astra Control Provisioner creerà una relazione SnapMirror con il nome della policy di relazione configurata (o default per ONTAP) e la inizializzerà.

c. Crea un PVC con StorageClass creato in precedenza per agire come secondario (destinazione SnapMirror).

Esempio

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Astra Control Provisioner controllerà la CRD TridentMirrorRelationship e non creerà il volume se la relazione non esiste. Se esiste una relazione, Astra Control Provisioner garantirà che il nuovo volume FlexVol venga inserito in una SVM a cui viene inviata la SVM remota definita in MirrorRelationship.

Stati di replica dei volumi

Una relazione mirror Trident (TMR) è un CRD che rappresenta un'estremità di una relazione di replica tra PVC. Il TMR di destinazione ha uno stato, che indica ad Astra Control Provisioner lo stato desiderato. Il TMR di destinazione ha i seguenti stati:

- **Stabilito**: Il PVC locale è il volume di destinazione di una relazione speculare, e questa è una nuova relazione.

- **Promosso**: Il PVC locale è ReadWrite e montabile, senza alcuna relazione speculare attualmente in vigore.
- **Ristabilito**: Il PVC locale è il volume di destinazione di una relazione speculare ed era anche precedentemente in quella relazione speculare.
  - Lo stato ristabilito deve essere utilizzato se il volume di destinazione era in una relazione con il volume di origine perché sovrascrive il contenuto del volume di destinazione.
  - Se il volume non era precedentemente in relazione con l'origine, lo stato ristabilito non riuscirà.

Promozione del PVC secondario durante un failover non pianificato

Eseguire il seguente passaggio sul cluster Kubernetes secondario:

- Aggiornare il campo *spec.state* di TridentMirrorRelationship a `promoted`.

Promozione del PVC secondario durante un failover pianificato

Durante un failover pianificato (migrazione), eseguire le seguenti operazioni per promuovere il PVC secondario:

Fasi

1. Sul cluster Kubernetes primario, creare una snapshot del PVC e attendere la creazione dello snapshot.
2. Sul cluster Kubernetes primario, creare SnapshotInfo CR per ottenere dettagli interni.

   Esempio

   ```
   kind: SnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshot-name: csi-nas-snapshot
   ```

3. Nel cluster Kubernetes secondario, aggiornare il campo *spec.state* del *TridentMirrorRelationship* CR a *Promoted* e *spec.promotedSnapshotHandle* come nome interno dello snapshot.
4. Sul cluster Kubernetes secondario, confermare lo stato (campo status.state) di TridentMirrorRelationship a promosso.

Ripristinare una relazione di mirroring dopo un failover

Prima di ripristinare una relazione di specchiatura, scegliere il lato che si desidera creare come nuovo primario.

Fasi

1. Nel cluster Kubernetes secondario, verificare che i valori per il campo *spec.remoteVolumeHandle* in TridentMirrorRelationship siano aggiornati.
2. Sul cluster Kubernetes secondario, aggiornare il campo *spec.mirror* di TridentMirrorRelationship a `reestablished`.

Operazioni supplementari

Astra Control Provivisioner supporta le seguenti operazioni sui volumi primario e secondario:

Replicare il PVC primario in un nuovo PVC secondario

Assicurarsi di disporre già di un PVC primario e di un PVC secondario.

Fasi

1. Eliminare i CRD PersistentVolumeClaim e TridentMirrorRelationship dal cluster (destinazione) secondario stabilito.
2. Eliminare il CRD TridentMirrorRelationship dal cluster primario (origine).
3. Creare un nuovo CRD TridentMirrorRelationship nel cluster primario (di origine) per il nuovo PVC secondario (di destinazione) che si desidera stabilire.

Ridimensionare un PVC specchiato, primario o secondario

Il PVC può essere ridimensionato normalmente, ONTAP espanderà automaticamente qualsiasi flevxols di destinazione se la quantità di dati supera le dimensioni correnti.

Rimuovere la replica da un PVC

Per rimuovere la replica, eseguire una delle seguenti operazioni sul volume secondario corrente:

- Eliminare MirrorRelationship sul PVC secondario. Questo interrompe la relazione di replica.
- In alternativa, aggiornare il campo spec.state a *Promoted*.

Eliminazione di un PVC (precedentemente specchiato)

Astra Control Provivisioner verifica la presenza di PVC replicati e rilascia la relazione di replica prima di tentare di eliminare il volume.

Eliminare una TMR

L'eliminazione di una TMR su un lato di una relazione specchiata fa sì che la TMR rimanente passi allo stato *promosso* prima che Astra Control Provivisioner completi l'eliminazione. Se il TMR selezionato per l'eliminazione è già nello stato *promosso*, non esiste alcuna relazione di mirror esistente e il TMR verrà rimosso e Astra Control Provisioner promuoverà il PVC locale in *ReadWrite*. Questa eliminazione rilascia i metadati SnapMirror per il volume locale in ONTAP. Se in futuro questo volume viene utilizzato in una relazione di mirroring, deve utilizzare un nuovo TMR con uno stato di replica del volume *stabilito* quando si crea la nuova relazione di mirroring.

Aggiorna relazioni mirror quando ONTAP è online

Le relazioni speculari possono essere aggiornate in qualsiasi momento dopo che sono state stabilite. È possibile utilizzare i `state: promoted` campi o `state: reestablished` per aggiornare le relazioni. Quando si trasferisce un volume di destinazione a un volume ReadWrite regolare, è possibile utilizzare *PromotedSnapshotHandle* per specificare uno snapshot specifico su cui ripristinare il volume corrente.

Aggiorna relazioni di mirroring quando ONTAP non è in linea

Puoi utilizzare un CRD per eseguire un update di SnapMirror senza che Astra Control disponga di connettività diretta al cluster ONTAP. Fare riferimento al seguente formato di esempio di TridentActionMirrorUpdate:

Esempio

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` Riflette lo stato del CRD TridentActionMirrorUpdate. Può assumere un valore da *riuscito*, *in corso* o *non riuscito*.

Abilita Astra Control Provisioner

Le versioni 23,10 e successive di Trident includono la possibilità di utilizzare Astra Control Provisioner, che consente agli utenti dotati di licenza Astra Control di accedere a funzionalità avanzate di provisioning dello storage. Astra Control Provisioner fornisce questa funzionalità estesa oltre alle funzionalità standard basate su CSI Astra Trident. È possibile utilizzare questa procedura per abilitare e installare Astra Control Provisioner.

L'abbonamento a Astra Control Service include automaticamente la licenza per l'utilizzo di Astra Control Provisioner.

In arrivo gli update di Astra Control, Astra Control Provivisioner sostituirà Astra Trident come provisioner di storage e orchestrator e sarà obbligatorio per l'utilizzo di Astra Control. Per questo motivo, si consiglia vivamente agli utenti di Astra Control di attivare Astra Control Provisioner. Astra Trident continuerà a rimanere open source e ad essere rilasciato, mantenuto, supportato e aggiornato con le nuove CSI e altre funzionalità di NetApp.

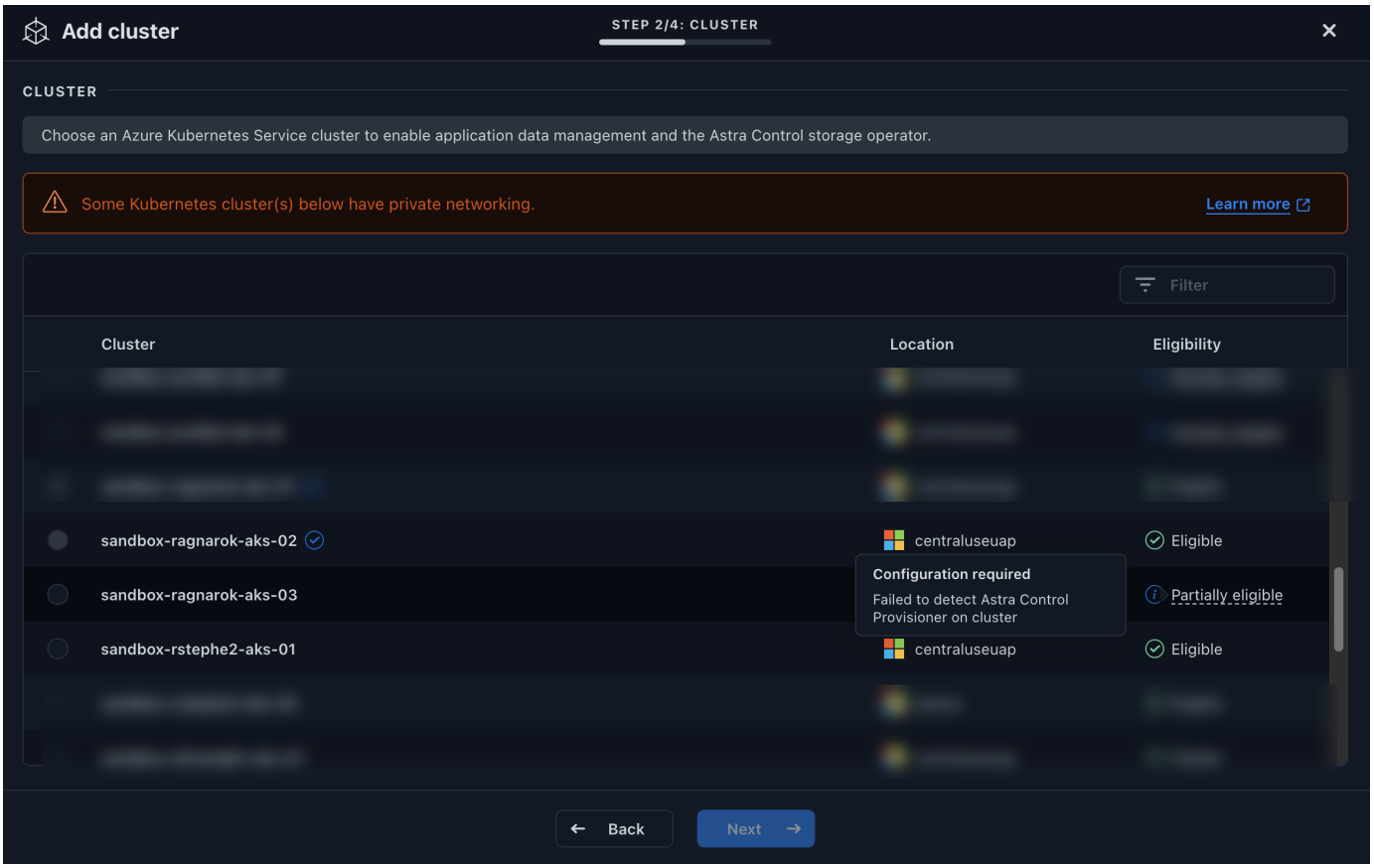
Come faccio a sapere se devo abilitare Astra Control Provivisioner?

Se si aggiunge un cluster ad Astra Control Service che non ha Astra Trident precedentemente installato, il cluster verrà contrassegnato come `Eligible`. Dopo di voi "Aggiungi il cluster a Astra Control", Astra Control Provivisioner verrà abilitato automaticamente.

Se il cluster non è contrassegnato `Eligible`, verrà contrassegnato `Partially eligible` a causa di uno dei seguenti fattori:

- Sta utilizzando una versione meno recente di Astra Trident
- Sta utilizzando un Astra Trident 23,10 che non ha ancora attivato l'opzione di provisioning
- Si tratta di un tipo di cluster che non consente l'abilitazione automatica

Per `Partially eligible` i casi, usa queste istruzioni per abilitare manualmente Astra Control Provisioner per il tuo cluster.



Prima di attivare Astra Control Provisioner

Se hai già un Astra Trident senza Astra Control Provivisioner e vuoi abilitare Astra Control Provivisioner, esegui prima quanto segue:

- **Se Astra Trident è installato, conferma che la sua versione si trovi all'interno di una finestra a quattro release**: Puoi eseguire un aggiornamento diretto a Astra Trident 24,02 con Astra Control Provisioner se il tuo Astra Trident si trova all'interno di una finestra a quattro release della versione 24,02. Ad esempio, puoi eseguire l'upgrade direttamente da Astra Trident 23,04 a 24,02.
- **Confirmi che il tuo cluster ha un'architettura di sistema AMD64**: L'immagine Astra Control Provivisioner è fornita in entrambe le architetture CPU AMD64 e ARM64, ma solo AMD64 è supportato da Astra Control.

Fasi

1. Accedere al Registro di sistema dell'immagine di controllo Astra di NetApp:
   a. Effettua l'accesso all'interfaccia utente di Astra Control Service e registra l'ID account Astra Control.
      i. Selezionare l'icona della figura in alto a destra nella pagina.
      ii. Selezionare **API access**.
      iii. Annotare l'ID account.
   b. Nella stessa pagina, selezionare **generate API token**, copiare la stringa del token API negli Appunti e salvarla nell'editor.
   c. Accedere al registro Astra Control utilizzando il metodo preferito:

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

```
crane auth login cr.astra.netapp.io -u <account-id> -p <api-token>
```

2. (Solo registri personalizzati) attenersi alla seguente procedura per spostare l'immagine nel registro personalizzato. Se non si utilizza un registro, seguire i passaggi dell'operatore Trident nella sezione successiva.

   Per i seguenti comandi, puoi utilizzare Podman al posto di Docker. Se si utilizza un ambiente Windows, si consiglia di utilizzare PowerShell.

   Docker

   a. Estrarre l'immagine di Astra Control provisioner dal Registro di sistema:

      L'immagine estratta non supporta più piattaforme e supporta solo la stessa piattaforma dell'host che ha estratto l'immagine, ad esempio Linux AMD64.

      ```
      docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0
      --platform <cluster platform>
      ```

      Esempio:

      ```
      docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0
      --platform linux/amd64
      ```

   b. Contrassegnare l'immagine:

      ```
      docker tag cr.astra.netapp.io/astra/trident-acp:24.02.0
      <my_custom_registry>/trident-acp:24.02.0
      ```

   c. Inviare l'immagine al registro personalizzato:

      ```
      docker push <my_custom_registry>/trident-acp:24.02.0
      ```

   Gru

   a. Copiare il manifesto di Astra Control Provisioner nel registro personalizzato:

      ```
      crane copy cr.astra.netapp.io/astra/trident-acp:24.02.0
      <my_custom_registry>/trident-acp:24.02.0
      ```

3. Determinare se il metodo di installazione originale di Astra Trident ha utilizzato un.

4. Abilita Astra Control Provisioner in Astra Trident utilizzando il metodo di installazione utilizzato originariamente:

Operatore Astra Trident

a. "Scaricare il programma di installazione di Astra Trident ed estrarlo".

b. Completa questi passaggi se non hai ancora installato Astra Trident o se hai rimosso l'operatore dall'implementazione originale di Astra Trident:

   i. Creare il CRD:

   ```
   kubectl create -f
   deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.1
   6.yaml
   ```

   ii. Creare lo spazio dei nomi tridente (`kubectl create namespace trident`) o confermare che lo spazio dei nomi tridente esista ancora (`kubectl get all -n trident`). Se lo spazio dei nomi è stato rimosso, crearlo di nuovo.

c. Aggiorna Astra Trident alla versione 24.02.0:

   Per i cluster che eseguono Kubernetes 1,24 o versione precedente, utilizzare `bundle_pre_1_25.yaml`. Per i cluster che eseguono Kubernetes 1,25 o versione successiva, utilizzare `bundle_post_1_25.yaml`.

   ```
   kubectl -n trident apply -f trident-installer/deploy/<bundle-
   name.yaml>
   ```

d. Verificare che Astra Trident sia in esecuzione:

   ```
   kubectl get torc -n trident
   ```

   Risposta:

   ```
   NAME      AGE
   trident   21m
   ```

e. se si dispone di un registro che utilizza segreti, creare un segreto da utilizzare per estrarre l'immagine di Astra Control Provisioner:

   ```
   kubectl create secret docker-registry <secret_name> -n trident
   --docker-server=<my_custom_registry> --docker-username=<username>
   --docker-password=<token>
   ```

f. Modificare il TridentOrchestrator CR e apportare le seguenti modifiche:

```
kubectl edit torc trident -n trident
```

i. Impostare una posizione del Registro di sistema personalizzata per l'immagine Astra Trident o estrarla dal Registro di sistema Astra Control (`tridentImage: <my_custom_registry>/trident:24.02.0` o `tridentImage: netapp/trident:24.02.0`).

ii. Attiva Astra Control Provivioner (`enableACP: true`).

iii. Impostare la posizione del Registro di sistema personalizzata per l'immagine Astra Control Provivioner o estrarla dal Registro di sistema Astra Control (`acpImage: <my_custom_registry>/trident-acp:24.02.0` o `acpImage: cr.astra.netapp.io/astra/trident-acp:24.02.0`).

iv. Se è stato stabilito segreti di estrazione delle immagini precedentemente in questa procedura, è possibile impostarli qui (`imagePullSecrets: - <secret_name>`). Usare lo stesso nome segreto che hai stabilito nei passaggi precedenti.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  tridentImage: <registry>/trident:24.02.0
  enableACP: true
  acpImage: <registry>/trident-acp:24.02.0
  imagePullSecrets:
  - <secret_name>
```

g. Salvare e uscire dal file. Il processo di distribuzione si avvia automaticamente.

h. Verificare che l'operatore, la distribuzione e i replicaset siano stati creati.

```
kubectl get all -n trident
```

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

i. Verificare che il `trident-acp` contenitore sia in funzione e che `acpVersion` lo stato sia `24.02.0 Installed`:

```
kubectl get torc -o yaml
```

Risposta:

```
status:
  acpVersion: 24.02.0
  currentInstallationParams:
    ...
    acpImage: <registry>/trident-acp:24.02.0
    enableACP: "true"
    ...
  ...
  status: Installed
```

tridentctl

a. "Scaricare il programma di installazione di Astra Trident ed estrarlo".

b. "Se disponi già di un Astra Trident, disinstallarlo dal cluster che lo ospita".

c. Installare Astra Trident con Astra Control Provivisioner abilitato (`--enable-acp=true`):

```
./tridentctl -n trident install --enable-acp=true --acp
-image=mycustomregistry/trident-acp:24.02
```

d. Confermare che Astra Control Provisioner è stato abilitato:

```
./tridentctl -n trident version
```

Risposta:

```
+----------------+----------------+-------------+ | SERVER
VERSION | CLIENT VERSION | ACP VERSION | +----------------
+----------------+-------------+ | 24.02.0 | 24.02.0 | 24.02.0. |
+----------------+----------------+-------------+
```

Timone

a. Se Astra Trident 23.07.1 o versione precedente è installato, "disinstallazione" l'operatore e altri componenti.

b. Se il cluster Kubernetes esegue la versione 1,24 o precedente, elimina psp:

```
kubectl delete psp tridentoperatorpod
```

c. Aggiungere il repository Astra Trident Helm:

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

d. Aggiornare il grafico Helm:

```
helm repo update netapp-trident
```

Risposta:

```
Hang tight while we grab the latest from your chart
repositories...
...Successfully got an update from the "netapp-trident" chart
repository
Update Complete. □Happy Helming!□
```

e. Elencare le immagini:

```
./tridentctl images -n trident
```

Risposta:

```
| v1.28.0            | netapp/trident:24.02.0|
|                    | docker.io/netapp/trident-
autosupport:24.02|
|                    | registry.k8s.io/sig-storage/csi-
provisioner:v4.0.0|
|                    | registry.k8s.io/sig-storage/csi-
attacher:v4.5.0|
|                    | registry.k8s.io/sig-storage/csi-
resizer:v1.9.3|
|                    | registry.k8s.io/sig-storage/csi-
snapshotter:v6.3.3|
|                    | registry.k8s.io/sig-storage/csi-node-
driver-registrar:v2.10.0 |
|                    | netapp/trident-operator:24.02.0 (optional)
```

f. Assicurarsi che l'operatore di tridente 24.02.0 sia disponibile:

```
helm search repo netapp-trident/trident-operator --versions
```

Risposta:

```
NAME                                CHART VERSION   APP VERSION
DESCRIPTION
netapp-trident/trident-operator 100.2402.0      24.02.0         A
```

g. Utilizzare `helm install` ed eseguire una delle seguenti opzioni che includono queste impostazioni:

- Un nome per la posizione di distribuzione
- La versione di Astra Trident
- Il nome dell'immagine di Astra Control provisioner
- Il flag per abilitare il provisioner
- (Facoltativo) percorso del Registro di sistema locale. Se si utilizza un registro locale, lo "Immagini Trident" può trovarsi in un registro o in registri diversi, ma tutte le immagini CSI devono trovarsi nello stesso registro.
- Il namespace Trident

Opzioni

◦ Immagini senza registro

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=cr.astra.netapp.io/astra/trident-
acp:24.02.0 --set enableACP=true --set operatorImage=netapp/trident-
operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

◦ Immagini in uno o più registri

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=<your-registry>:<acp image> --set
enableACP=true --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=netapp/trident-operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

È possibile utilizzare `helm list` per rivedere i dettagli dell'installazione, ad esempio nome, spazio dei nomi, grafico, stato, versione dell'applicazione, e numero di revisione.

Se hai problemi nell'implementazione di Trident utilizzando Helm, esegui questo comando per disinstallare completamente Astra Trident:

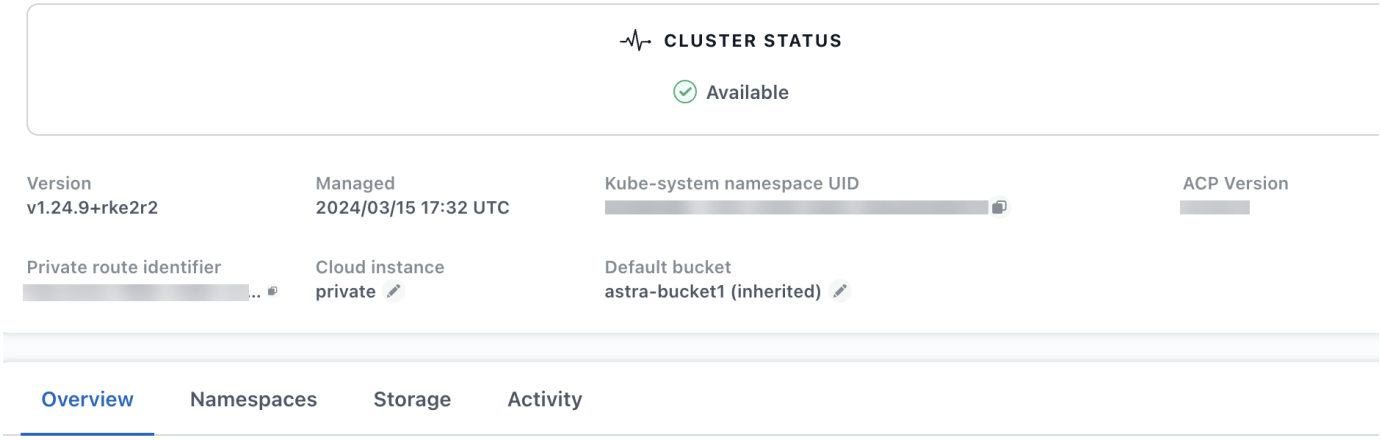
```
./tridentctl uninstall -n trident
```

Non "Rimuovere completamente i CRD Astra Trident" come parte della disinstallazione prima di tentare di abilitare nuovamente Astra Control Provisioner.

Risultato

La funzionalità Astra Control Provisioner è abilitata ed è possibile utilizzare qualsiasi funzionalità disponibile per la versione in esecuzione.

Dopo l'installazione di Astra Control provisioner, il cluster che ospita il provisioner nell'interfaccia utente di Astra Control mostrerà un `ACP version` numero di versione installata piuttosto che `Trident version` sul campo.



Per ulteriori informazioni

- "Documentazione sugli aggiornamenti di Astra Trident"

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".

Panoramica

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 che esegue un "Versione Kubernetes supportata"

```
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.

Ecco un esempio di definizione di backend:

YAML

```
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
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
```

JSON

```
{
 "version": 1,
 "storageDriverName": "ontap-san",
 "backendName": "san-backend-us-east1",
 "managementLIF": "192.168.27.5",
 "svm": "iscsi_svm",
 "username": "admin",
 "password": "password",
 "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: password
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

Le snapshot del volume di Kubernetes dei volumi persistenti (PVS) consentono copie point-in-time dei volumi. Puoi creare una snapshot di un volume creato utilizzando Astra Trident, importare uno snapshot creato all'esterno di Astra Trident, creare un nuovo volume da una snapshot esistente e recuperare i dati del volume da snapshot.

Panoramica

Lo snapshot del volume è supportato da `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, e. `azure-netapp-files` driver.

Prima di iniziare

Per utilizzare gli snapshot, è necessario disporre di un controller snapshot esterno e di CRD (Custom Resource Definitions). Questa è la responsabilità del Kubernetes orchestrator (ad esempio: Kubeadm, GKE, OpenShift).

Se la distribuzione Kubernetes non include il controller di snapshot e i CRD, fare riferimento a. Implementare un controller per lo snapshot dei volumi.

> Non creare un controller di snapshot se si creano snapshot di volumi on-demand in un ambiente GKE. GKE utilizza un controller di snapshot integrato e nascosto.

Creare un'istantanea del volume

Fasi

1. Creare un `VolumeSnapshotClass`. Per ulteriori informazioni, fare riferimento a. "VolumeSnapshotClass".
   - Il `driver` Indica il driver Astra Trident CSI.
   - `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.

     Esempio

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

2. Creare un'istantanea di un PVC esistente.

   Esempi

   - Questo esempio crea un'istantanea di un PVC esistente.

     ```
     cat snap.yaml
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshot
     metadata:
       name: pvc1-snap
     spec:
       volumeSnapshotClassName: csi-snapclass
       source:
         persistentVolumeClaimName: pvc1
     ```

   - Questo esempio crea un oggetto snapshot di volume per un PVC denominato `pvc1` e il nome dello snapshot è impostato su `pvc1-snap`. Un'istantanea VolumeSnapshot è analoga a un PVC ed è associata a un `VolumeSnapshotContent` oggetto che rappresenta lo snapshot effettivo.

     ```
     kubectl create -f snap.yaml
     volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

     kubectl get volumesnapshots
     NAME                    AGE
     pvc1-snap               50s
     ```

- È possibile identificare `VolumeSnapshotContent` oggetto per `pvc1-snap` VolumeSnapshot descrivendolo. Il `Snapshot Content Name` Identifica l'oggetto VolumeSnapshotContent che fornisce questa snapshot. Il `Ready To Use` Parametro indica che l'istantanea può essere utilizzata per creare un nuovo PVC.

```
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
.
.
```

Creare un PVC da uno snapshot di volume

È possibile utilizzare `dataSource` Per creare un PVC utilizzando un VolumeSnapshot denominato `<pvc-name>` come origine dei dati. Una volta creato, il PVC può essere collegato a un pod e utilizzato come qualsiasi altro PVC.

> Il PVC verrà creato nello stesso backend del volume di origine. Fare riferimento a. "KB: La creazione di un PVC da uno snapshot PVC Trident non può essere creata in un backend alternativo".

Nell'esempio seguente viene creato il PVC utilizzando `pvc1-snap` come origine dei dati.

```
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
```

Importare uno snapshot di volume

Astra Trident supporta a. "Processo Snapshot con pre-provisioning di Kubernetes" per consentire all'amministratore del cluster di creare un `VolumeSnapshotContent` Object e importa gli snapshot creati all'esterno di Astra Trident.

Prima di iniziare

Astra Trident deve aver creato o importato il volume principale dello snapshot.

Fasi

1. **Cluster admin:** creare un `VolumeSnapshotContent` oggetto che fa riferimento allo snapshot backend. In questo modo viene avviato il flusso di lavoro delle snapshot in Astra Trident.
   - Specificare il nome dell'istantanea backend in `annotations` come `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
   - Specificare `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` poll `snapshotHandle`. Queste sono le uniche informazioni fornite a Astra Trident dallo snap-ter esterno in `ListSnapshots` chiamata.

     > ⓘ Il `<volumeSnapshotContentName>` Impossibile corrispondere sempre al nome dell'istantanea backend a causa di vincoli di denominazione CR.

     Esempio

     Nell'esempio seguente viene creato un `VolumeSnapshotContent` oggetto che fa riferimento allo snapshot backend `snap-01`.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
```

2. **Cluster admin:** creare il `VolumeSnapshot` CR che fa riferimento a. `VolumeSnapshotContent` oggetto. In questo modo viene richiesto l'accesso per l'utilizzo di `VolumeSnapshot` in un determinato namespace.

   Esempio

   Nell'esempio seguente viene creato un `VolumeSnapshot` CR con nome `import-snap` questo fa riferimento al `VolumeSnapshotContent` con nome `import-snap-content`.

   ```
   apiVersion: snapshot.storage.k8s.io/v1
   kind: VolumeSnapshot
   metadata:
     name: import-snap
   spec:
     # volumeSnapshotClassName: csi-snapclass (not required for pre-
   provisioned or imported snapshots)
     source:
       volumeSnapshotContentName: import-snap-content
   ```

3. **Elaborazione interna (nessuna azione richiesta):** lo snapshot esterno riconosce la nuova creazione `VolumeSnapshotContent` ed esegue `ListSnapshots` chiamata. Astra Trident crea l' `TridentSnapshot`.
   - Lo snapshot esterno imposta `VolumeSnapshotContent` a. `readyToUse` e a. `VolumeSnapshot` a. `true`.
   - Trident ritorna `readyToUse=true`.
4. **Qualsiasi utente:** creare un `PersistentVolumeClaim` per fare riferimento al nuovo `VolumeSnapshot`, dove il `spec.dataSource` (o. `spec.dataSourceRef`) è il `VolumeSnapshot` nome.

   Esempio

   Nell'esempio seguente viene creato un PVC che fa riferimento a. `VolumeSnapshot` con nome `import-snap`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Ripristinare i dati del volume utilizzando le snapshot

La directory Snapshot è nascosta per impostazione predefinita per facilitare la massima compatibilità dei volumi con cui viene eseguito il provisioning mediante `ontap-nas` e. `ontap-nas-economy` driver. Attivare il `.snapshot` directory per ripristinare i dati direttamente dalle snapshot.

Utilizzare la CLI ONTAP per il ripristino dello snapshot del volume per ripristinare uno stato di un volume registrato in uno snapshot precedente.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

> Quando si ripristina una copia snapshot, la configurazione del volume esistente viene sovrascritta. Le modifiche apportate ai dati del volume dopo la creazione della copia snapshot andranno perse.

La directory Snapshot è nascosta per impostazione predefinita per facilitare la massima compatibilità dei volumi con cui viene eseguito il provisioning mediante `ontap-nas` e. `ontap-nas-economy` driver. Attivare il `.snapshot` directory per ripristinare i dati direttamente dalle snapshot.

> Quando si ripristina una copia snapshot, la configurazione del volume esistente viene sovrascritta. Le modifiche apportate ai dati del volume dopo la creazione della copia snapshot andranno perse.

Ripristino del volume in-place da uno snapshot

Astra Control Provisioner consente il ripristino rapido e in-place dei volumi da uno snapshot utilizzando il `TridentActionSnapshotRestore` CR (TASR). Questo CR funziona come un'azione imperativa di Kubernetes e non persiste al termine dell'operazione.

Astra Control Provivisioner supporta il ripristino delle istantanee su `ontap-san`, `ontap-san-economy` `ontap-nas`, , `ontap-nas-flexgroup`, , `azure-netapp-files` `gcp-cvs`, e `solidfire-san` driver.

Prima di iniziare

È necessario disporre di un PVC associato e di uno snapshot del volume disponibile.

- Verificare che lo stato del PVC sia limitato.

```
kubectl get pvc
```

- Verificare che lo snapshot del volume sia pronto per l'uso.

```
kubectl get vs
```

Fasi

1. Creare TASR CR. In questo esempio viene creata una CR per PVC `pvc1` e snapshot volume `pvc1-snapshot`.

```
cat tasr-pvc1-snapshot.yaml

apiVersion: v1
kind: TridentActionSnapshotRestore
metadata:
  name: this-doesnt-matter
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. Applicare la CR per eseguire il ripristino dall'istantanea. Nell'esempio riportato di seguito vengono ripristinati gli snapshot `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml

tridentactionsnapshotrestore.trident.netapp.io/this-doesnt-matter
created
```

Risultati

Astra Control Provivisioner ripristina i dati dalla snapshot. È possibile verificare lo stato di ripristino dello snapshot.

```
kubectl get tasr -o yaml

apiVersion: v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: this-doesnt-matter
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

- Nella maggior parte dei casi, Astra Control provisioner non ritenta automaticamente l'operazione in caso di guasto. Sarà necessario eseguire nuovamente l'operazione.
- Gli utenti Kubernetes senza accesso amministrativo potrebbero dover essere autorizzati dall'amministratore a creare una TASR CR nel namespace delle applicazioni.

Utilizzare la CLI ONTAP per il ripristino dello snapshot del volume per ripristinare uno stato di un volume registrato in uno snapshot precedente.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

Eliminare un PV con gli snapshot associati

Quando si elimina un volume persistente con snapshot associate, il volume Trident corrispondente viene aggiornato a uno stato di eliminazione. Rimuovere le snapshot del volume per eliminare il volume Astra Trident.

Implementare un controller per lo snapshot dei volumi

Se la distribuzione Kubernetes non include lo snapshot controller e i CRD, è possibile implementarli come segue.

Fasi

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-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Creare il controller di snapshot.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

> Se necessario, aprire `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e aggiornare `namespace` allo spazio dei nomi.

Link correlati

- "Snapshot dei volumi"
- "VolumeSnapshotClass"

Gestisci e monitora Astra Trident

Aggiorna Astra Trident

Aggiorna Astra Trident

A partire dalla release 24,02, Astra Trident segue una cadenza di quattro mesi, fornendo tre release principali ogni anno solare. Ogni nuova release si basa sulle release precedenti e offre nuove funzionalità, miglioramenti delle prestazioni, correzioni di bug e miglioramenti. Ti consigliamo di effettuare l'upgrade almeno una volta all'anno per sfruttare le nuove funzionalità di Astra Trident.

Considerazioni prima dell'aggiornamento

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

- Dovrebbe essere installata una sola istanza di Astra Trident in tutti gli spazi dei nomi di un determinato cluster Kubernetes.
- Astra Trident 23,07 e versioni successive richiede snapshot di volume v1 e non supporta più snapshot alfa o beta.
- Se hai creato Cloud Volumes Service per Google Cloud nel "Tipo di servizio CVS", è necessario aggiornare la configurazione del backend per utilizzare `standardsw` oppure `zoneredundantstandardsw` Livello di servizio durante l'upgrade da Astra Trident 23,01. Impossibile aggiornare `serviceLevel` nel backend potrebbe causare un errore nei volumi. Fare riferimento a. "Esempi di tipo di servizio CVS" per ulteriori informazioni.
- 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 "contesti di sicurezza" Per volumi SAN.
  - La directory sample input contiene esempi, come `storage-class-basic.yaml.templ` e `storage-class-bronze-default.yaml`.
  - Per ulteriori informazioni, fare riferimento a. "Problemi noti".

Fase 1: Selezionare una versione

Le versioni di Astra Trident seguono una data-based `YY.MM` Convenzione di naming, dove "YY" è l'ultima cifra dell'anno e "MM" è il mese. I rilasci di punti seguono un `YY.MM.X` convention, dove "X" è il livello di patch. Selezionare la versione a cui eseguire l'aggiornamento in base alla versione da cui si sta eseguendo l'aggiornamento.

- È possibile eseguire un aggiornamento diretto a qualsiasi release di destinazione che si trova all'interno di una finestra di quattro release della versione installata. Ad esempio, è possibile aggiornare direttamente da 23,01 (o qualsiasi versione a 23,01 punti) a 24,02.
- Se si sta eseguendo l'aggiornamento da una release al di fuori della finestra a quattro release, eseguire un aggiornamento in più fasi. Seguire le istruzioni di aggiornamento di "versione precedente" si sta eseguendo l'aggiornamento alla versione più recente che si adatta alla finestra a quattro release. Ad esempio, se si utilizza 22,01 e si desidera eseguire l'aggiornamento a 24,02:
  a. Primo aggiornamento da 22,01 a 23,01.

b. Quindi, eseguire l'aggiornamento da 23,01 a 24,02.

> Quando si esegue l'aggiornamento utilizzando l'operatore Trident 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, fare riferimento alla "Dettagli del problema su GitHub".

Fase 2: Determinare il metodo di installazione originale

Per determinare la versione utilizzata per installare Astra Trident:

1. Utilizzare `kubectl get pods -n trident` esaminare i pod.
   - Se non è disponibile un pod per l'operatore, Astra Trident è stato installato utilizzando `tridentctl`.
   - Se è presente un pod operatore, Astra Trident è stato installato utilizzando l'operatore Trident manualmente o utilizzando Helm.
2. Se è presente un pod per l'operatore, utilizzare `kubectl describe torc` Per determinare se Astra Trident è stato installato utilizzando Helm.
   - Se è presente un'etichetta Helm, Astra Trident è stato installato utilizzando Helm.
   - Se non è presente alcuna etichetta Helm, Astra Trident è stato installato manualmente utilizzando l'operatore Trident.

Fase 3: Selezionare un metodo di aggiornamento

In genere, è necessario eseguire l'aggiornamento utilizzando lo stesso metodo utilizzato per l'installazione iniziale, tuttavia è possibile "passare da un metodo di installazione all'altro". Ci sono due opzioni per aggiornare Astra Trident.

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

> Ti consigliamo di rivedere "Comprendere il flusso di lavoro di aggiornamento dell'operatore" prima di eseguire l'aggiornamento con l'operatore.

*

Eseguire l'upgrade con l'operatore

Comprendere il flusso di lavoro di aggiornamento dell'operatore

Prima di utilizzare l'operatore Trident per aggiornare Astra Trident, devi comprendere i processi in background che si verificano durante l'upgrade. Sono incluse le modifiche al controller Trident, al pod dei controller e ai pod dei nodi e al daemonSet dei nodi che consentono l'esecuzione degli aggiornamenti.

Gestione dell'aggiornamento dell'operatore Trident

Uno dei tanti "Vantaggi dell'utilizzo dell'operatore Trident" Per installare e aggiornare Astra Trident è la gestione automatica degli oggetti Astra Trident e Kubernetes senza interrompere i volumi montati. In questo modo, Astra Trident può supportare gli upgrade senza downtime o. "*rolling updates*". In particolare, l'operatore Trident comunica con il cluster Kubernetes per:

- Eliminare e ricreare l'implementazione del controller Trident e il daemonSet del nodo.
- Sostituisci il Controller Pod Trident e i pod di nodi Trident con nuove versioni.
  - Se un nodo non viene aggiornato, non impedisce l'aggiornamento dei nodi rimanenti.
  - Solo i nodi con un pod nodo Trident in esecuzione possono montare volumi.

Per ulteriori informazioni sull'architettura Astra Trident nel cluster Kubernetes, fare riferimento a. "Architettura Astra Trident".

Flusso di lavoro di aggiornamento dell'operatore

Quando si avvia un aggiornamento utilizzando l'operatore Trident:

1. L'operatore **Trident**:
   a. Rileva la versione attualmente installata di Astra Trident (versione *n*).
   b. Aggiorna tutti gli oggetti Kubernetes, inclusi CRD, RBAC e Trident SVC.
   c. Elimina l'implementazione del controller Trident per la versione *n*.
   d. Crea l'implementazione del controller Trident per la versione *n+1*.
2. **Kubernetes** crea il Pod controller Trident per *n+1*.
3. L'operatore **Trident**:
   a. Elimina il daemonSet del nodo Trident per *n*. L'operatore non attende la terminazione del nodo Pod.
   b. Crea il nodo Trident Daemonset per *n+1*.
4. **Kubernetes** crea pod di nodi Trident sui nodi che non eseguono il pod di nodi Trident *n*. In questo modo, si garantisce che non ci sia mai più di un Pod nodi Trident, di qualsiasi versione, su un nodo.

Aggiornare un'installazione Astra Trident usando l'operatore Trident o Helm

È possibile eseguire l'aggiornamento di Astra Trident utilizzando l'operatore Trident sia manualmente che tramite Helm. È possibile eseguire l'aggiornamento da un'installazione dell'operatore Trident a un'altra installazione dell'operatore Trident o da un `tridentctl` Installazione su una versione dell'operatore Trident. Revisione "Selezionare un metodo di aggiornamento" Prima di aggiornare un'installazione dell'operatore Trident.

Aggiornare un'installazione manuale

È possibile eseguire l'aggiornamento da un'installazione dell'operatore Trident definita dall'ambito del cluster a un'altra installazione dell'operatore Trident definita dal cluster. Tutte le versioni di Astra Trident 21.01 e successive utilizzano un operatore con ambito cluster.

Per l'aggiornamento da Astra Trident che è stato installato usando l'operatore con ambito namespace (versioni da 20,07 a 20,10), usa le istruzioni di upgrade per "versione installata" Di Astra Trident.

A proposito di questa attività

Trident fornisce un file bundle da utilizzare per installare l'operatore e creare oggetti associati per la versione di Kubernetes.

- Per i cluster che eseguono Kubernetes 1.24 o versioni precedenti, utilizzare "bundle_pre_1_25.yaml".
- Per i cluster che eseguono Kubernetes 1.25 o versioni successive, utilizzare "bundle_post_1_25.yaml".

Prima di iniziare

Assicurarsi di utilizzare un cluster Kubernetes in esecuzione "Una versione di Kubernetes supportata".

Fasi

1. Verificare la versione di Astra Trident:

```
./tridentctl -n trident version
```

2. Eliminare l'operatore Trident utilizzato per installare l'istanza corrente di Astra Trident. Ad esempio, se si sta eseguendo l'aggiornamento da 23,07, eseguire il seguente comando:

```
kubectl delete -f 23.07.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

3. Se l'installazione iniziale è stata personalizzata utilizzando `TridentOrchestrator` è possibile modificare `TridentOrchestrator` oggetto per modificare i parametri di installazione. Ciò potrebbe includere le modifiche apportate per specificare i registri di immagini Trident e CSI mirrorati per la modalità offline, abilitare i registri di debug o specificare i segreti di pull delle immagini.
4. Installare Astra Trident utilizzando il file YAML del bundle corretto per il proprio ambiente, dove *<bundle.yaml>* è `bundle_pre_1_25.yaml` oppure `bundle_post_1_25.yaml` In base alla versione di Kubernetes. Ad esempio, se stai installando Astra Trident 24,02, esegui il seguente comando:

```
kubectl create -f 24.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

Aggiornare un'installazione Helm

È possibile aggiornare un'installazione di Astra Trident Helm.

> Quando si aggiorna un cluster Kubernetes dalla versione 1.24 alla 1.25 o successiva su cui è installato Astra Trident, è necessario aggiornare values.yaml per impostarlo `excludePodSecurityPolicy` a. `true` oppure aggiungi `--set excludePodSecurityPolicy=true` al `helm upgrade` prima di aggiornare il cluster.

Fasi

1. Se "Installato Astra Trident utilizzando Helm", è possibile utilizzare `helm upgrade trident netapp-trident/trident-operator --version 100.2402.0` per eseguire l'aggiornamento in un solo passaggio. Se non è stato aggiunto il repo Helm o non è possibile utilizzarlo per l'aggiornamento:
   a. Scarica la versione più recente di Astra Trident da "La sezione *Assets* su GitHub".
   b. Utilizzare `helm upgrade` comando dove `trident-operator-24.02.0.tgz` indica la versione alla quale si desidera eseguire l'aggiornamento.

```
helm upgrade <name> trident-operator-24.02.0.tgz
```

Se si impostano opzioni personalizzate durante l'installazione iniziale (ad esempio, se si specificano registri privati con mirroring per le immagini Trident e CSI), aggiungere il `helm upgrade` utilizzare `--set` per assicurarsi che tali opzioni siano incluse nel comando upgrade, altrimenti i valori torneranno ai valori predefiniti.

2. Eseguire `helm list` per verificare che la versione del grafico e dell'applicazione sia stata aggiornata. Eseguire `tridentctl logs` per esaminare eventuali messaggi di debug.

Aggiornamento da a. `tridentctl` Installazione all'operatore Trident

È possibile eseguire l'aggiornamento all'ultima versione dell'operatore Trident da un `tridentctl` installazione. I backend e i PVC esistenti saranno automaticamente disponibili.

Prima di passare da un metodo di installazione all'altro, vedere "Passaggio da un metodo di installazione all'altro".

Fasi

1. Scarica l'ultima release di Astra Trident.

```
# Download the release required [24.020.0]
mkdir 24.02.0
cd 24.02.0
wget
https://github.com/NetApp/trident/releases/download/v24.02.0/trident-
installer-24.02.0.tar.gz
tar -xf trident-installer-24.02.0.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 con ambito cluster nello stesso namespace.

```
kubectl create -f deploy/<bundle-name.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-controller-79df798bdc-m79dc   6/6     Running   0          150d
trident-node-linux-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.

```
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
```

5. Confermare che Trident è stato aggiornato alla versione prevista.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v24.02.0
```

Upgrade con tridentctl

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

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.

Prima di iniziare

Revisione "Selezionare un metodo di aggiornamento" prima di eseguire l'aggiornamento con `tridentctl`.

Fasi

1. Eseguire il comando di disinstallazione in `tridentctl` Rimuovere tutte le risorse associate ad Astra Trident, ad eccezione dei CRD e degli oggetti correlati.

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

2. Reinstallare Astra Trident. Fare riferimento a. "Installare Astra Trident usando tridentctl".

> Non interrompere il processo di aggiornamento. Assicurarsi che il programma di installazione venga completato.

Gestisci Astra Trident usando tridentctl

Il "Pacchetto di installazione Trident" include il `tridentctl` Command-line utility per fornire un semplice accesso ad Astra Trident. Gli utenti Kubernetes con privilegi sufficienti possono usarlo per installare Astra Trident o gestire il namespace che contiene il pod Astra Trident.

Comandi e flag globali

Puoi correre `tridentctl help` per ottenere un elenco di comandi disponibili per `tridentctl` o aggiungere il `--help` flag a qualsiasi comando per ottenere un elenco di opzioni e flag per quel comando specifico.

```
tridentctl [command] [--optional-flag]
```

Astra Trident `tridentctl` utility supporta i seguenti comandi e flag globali.

Comandi

`create`
Aggiungi una risorsa a Astra Trident.

`delete`
Rimozione di una o più risorse da Astra Trident.

`get`
Ottieni una o più risorse da Astra Trident.

`help`
Aiuto su qualsiasi comando.

`images`
Stampare una tabella delle immagini container di cui Astra Trident ha bisogno.

`import`
Importa una risorsa esistente in Astra Trident.

`install`
Installa Astra Trident.

`logs`
Stampare i registri da Astra Trident.

`send`
Invia una risorsa da Astra Trident.

`uninstall`
Disinstallare Astra Trident.

`update`
Modifica una risorsa in Astra Trident.

`update backend state`
Sospendere temporaneamente le operazioni di backend.

`upgrade`
Aggiorna una risorsa in Astra Trident.

`version`
Stampa la versione di Astra Trident.

Flag globali

-d, --debug
Output di debug.

-h, --help
Aiuto per `tridentctl`.

-k, --kubeconfig string
Specificare `KUBECONFIG` Percorso per eseguire comandi in locale o da un cluster Kubernetes a un altro.

> In alternativa, è possibile esportare `KUBECONFIG` Variabile che indica un problema e un cluster Kubernetes specifici `tridentctl` comandi a quel cluster.

-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 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).

Opzioni di comando e flag

creare

Utilizzare `create` Comando per aggiungere una risorsa ad Astra Trident.

```
tridentctl create [option]
```

Opzioni

`backend`: Aggiungi un backend ad Astra Trident.

eliminare

Utilizzare `delete` Comando per rimuovere una o più risorse da Astra Trident.

```
tridentctl delete [option]
```

Opzioni

`backend`: Eliminare uno o più backend di storage 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.

ottieni

Utilizzare `get` Comando per ottenere una o più risorse da Astra Trident.

```
tridentctl get [option]
```

Opzioni

`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.

Allarmi

`-h`, `--help`: Guida per i volumi.
`--parentOfSubordinate string`: Limita query al volume di origine subordinato.
`--subordinateOf string`: Limita la query alle subordinate del volume.

immagini

Utilizzare `images` Flag per stampare una tabella delle immagini container di cui Astra Trident ha bisogno.

```
tridentctl images [flags]
```

Allarmi

`-h`, `--help`: Guida per le immagini.
`-v`, `--k8s-version string`: Versione semantica del cluster Kubernetes.

importa volume

Utilizzare `import volume` Comando per importare un volume esistente in Astra Trident.

```
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.

installare

Utilizzare `install` Flag per installare Astra Trident.

```
tridentctl install [flags]
```

Allarmi

`--autosupport-image string`: L'immagine contenitore per la telemetria AutoSupport (predefinita "netapp/trident autosupport:<current-version>").
`--autosupport-proxy string`Indirizzo/porta di un proxy per l'invio di telemetria AutoSupport.
`--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`: Ignorare il timeout della richiesta HTTP per l'API REST del controller Trident (valore predefinito 1m30).
`--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.

registri

Utilizzare `logs` Flag per stampare i log da Astra Trident.

```
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.

invia

Utilizzare `send` Comando per inviare una risorsa da Astra Trident.

```
tridentctl send [option]
```

Opzioni

`autosupport`: Inviare un archivio AutoSupport a NetApp.

disinstallazione

Utilizzare `uninstall` Flag per disinstallare Astra Trident.

```
tridentctl uninstall [flags]
```

Allarmi

`-h, --help`: Guida per la disinstallazione.
`--silent`: Disattivare la maggior parte dell'output durante la disinstallazione.

aggiornamento

Utilizzare `update` Comando per modificare una risorsa in Astra Trident.

```
tridentctl update [option]
```

Opzioni

`backend`: Aggiornare un backend in Astra Trident.

aggiorna stato backend

Utilizzare `update backend state` comando per sospendere o riprendere le operazioni di backend.

```
tridentctl update backend state <backend-name> [flag]
```

Allarmi

`-h, --help`: Guida per lo stato backend.
`--user-state`: Impostare su `suspended` per sospendere le operazioni di backend. Impostare su `normal` per riprendere le operazioni di backend. Quando è impostato su `suspended`:

- `AddVolume`, `CloneVolume`, `Import Volume`, `ResizeVolume` sono in pausa.
- `PublishVolume`, `UnPublishVolume`, `CreateSnapshot`, `GetSnapshot`, `RestoreSnapshot`, `DeleteSnapshot`, `RemoveVolume`, `GetVolumeExternal`, `ReconcileNodeAccess` rimangono disponibili.

versione

Utilizzare `version` contrassegni per stampare la versione di `tridentctl` E il servizio Running Trident.

```
tridentctl version [flags]
```

Allarmi

`--client`: Solo versione client (non è richiesto alcun server).
`-h, --help`: Guida per la versione.

Monitorare Astra Trident

Astra Trident offre un set di endpoint di metriche Prometheus che è possibile utilizzare per monitorare le performance di Astra Trident.

Panoramica

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 la creazione di un ServiceMonitor per ottenere le metriche 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 caratteristiche di Astra Trident
metriche.

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. Fare riferimento a. "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 NetApp
  "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
  Che viene 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" che non è applicabile all'immagine contenitore Trident stessa. Puoi saperne di più sull'impegno di NetApp nei confronti della sicurezza e della 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:
    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.

Disinstallare Astra Trident

Devi utilizzare lo stesso metodo per disinstallare Astra Trident che hai utilizzato per installare Astra Trident.

A proposito di questa attività

- Se hai bisogno di una correzione per i bug osservati dopo un aggiornamento, problemi di dipendenza o un aggiornamento non riuscito o incompleto, devi disinstallare Astra Trident e reinstallare la versione precedente utilizzando le istruzioni specifiche per questo "versione". Questo è l'unico modo consigliato per *eseguire il downgrade* a una versione precedente.
- Per semplificare l'aggiornamento e la reinstallazione, la disinstallazione di Astra Trident non rimuove i CRD

o gli oggetti correlati creati da Astra Trident. Se devi rimuovere completamente Astra Trident e tutti i suoi dati, fai riferimento a. "Rimuovere completamente Astra Trident e i CRD".

Prima di iniziare

Se stai decommissionando i cluster Kubernetes, devi eliminare tutte le applicazioni che utilizzano volumi creati da Astra Trident prima della disinstallazione. In questo modo, si garantisce che i PVC non siano pubblicati sui nodi Kubernetes prima di essere eliminati.

Determinare il metodo di installazione originale

Utilizzare lo stesso metodo per disinstallare Astra Trident che è stato utilizzato per installarlo. Prima di disinstallare, verificare quale versione è stata utilizzata per installare originariamente Astra Trident.

1. Utilizzare `kubectl get pods -n trident` esaminare i pod.
   - Se non è disponibile un pod per l'operatore, Astra Trident è stato installato utilizzando `tridentctl`.
   - Se è presente un pod operatore, Astra Trident è stato installato utilizzando l'operatore Trident manualmente o utilizzando Helm.
2. Se è presente un pod per l'operatore, utilizzare `kubectl describe tproc trident` Per determinare se Astra Trident è stato installato utilizzando Helm.
   - Se è presente un'etichetta Helm, Astra Trident è stato installato utilizzando Helm.
   - Se non è presente alcuna etichetta Helm, Astra Trident è stato installato manualmente utilizzando l'operatore Trident.

Disinstallare un'installazione dell'operatore Trident

È possibile disinstallare manualmente un'installazione dell'operatore tridente o utilizzando Helm.

Disinstallare l'installazione manuale

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

1. **Modifica `TridentOrchestrator` CR e impostare il flag di disinstallazione**:

   ```
   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.

2. **Elimina `TridentOrchestrator`**: Rimuovendo `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.

   ```
   kubectl delete -f deploy/<bundle.yaml> -n <namespace>
   ```

Disinstallare l'installazione di 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 un `tridentctl` installazione

Utilizzare `uninstall` ingresso comando `tridentctl` Per rimuovere tutte le risorse associate a Astra Trident, ad eccezione dei CRD e degli oggetti correlati:

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

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 i requisiti

- 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.

Strumenti NFS

Installa gli strumenti NFS utilizzando i comandi del tuo sistema operativo.

RHEL 8+

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```

Riavviare i nodi di lavoro dopo aver installato gli strumenti NFS per evitare errori durante il collegamento dei volumi ai container.

Strumenti iSCSI

Installare gli strumenti iSCSI utilizzando i comandi del sistema operativo.

RHEL 8+

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
```

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
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

Strumenti NVMe

Installa gli strumenti NVMe utilizzando i comandi del tuo sistema operativo.

- NVMe richiede RHEL 9 o versione successiva.
- Se la versione del kernel del nodo Kubernetes è troppo vecchia o se il pacchetto NVMe non è disponibile per la versione del kernel in uso, potrebbe essere necessario aggiornare la versione del kernel del nodo a una versione con il pacchetto NVMe.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Implementare Astra Trident

Astra Trident per Docker offre un'integrazione diretta con l'ecosistema Docker per le piattaforme di 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.

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": "password",
         "aggregate": "aggr1"
     }
     EOF
     ```

2. Avviare Astra Trident utilizzando il sistema di plugin gestito. Sostituire `<version>` con la versione del plugin (xxx.xx.x) in uso.

```
docker plugin install --grant-all-permissions --alias netapp
netapp/trident-plugin:<version> 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/v24.10.0/trident-
installer-24.02.0.tar.gz
tar zxf trident-installer-24.02.0.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": "password",
    "aggregate": "aggr1"
}
EOF
```

2. Dopo aver posizionato il file binario e creato il 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 del 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 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, fare riferimento a. "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

```
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"
                }
            ]
        }
    }
]

docker volume create -d ontap-nas --name clonedVolume -o from=firstVolume
clonedVolume

docker volume rm clonedVolume
docker volume create -d ontap-nas --name volFromSnap -o from=firstVolume
-o fromSnapshot=hourly.2017-02-10_1505
volFromSnap

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.
`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, la clonazione di un database vuoto può consentire un notevole risparmio di tempo ma anche di poco spazio di storage, pertanto è preferibile suddividere immediatamente il clone.

Opzione	Descrizione
`encryption`	Abilitare NetApp Volume Encryption (NVE) sul nuovo volume; il valore predefinito è `false`. NVE deve essere concesso in licenza e abilitato sul cluster per utilizzare questa opzione. Se NAE è attivato sul backend, tutti i volumi forniti in Astra Trident saranno abilitati per NAE. Per ulteriori informazioni, fare riferimento a: "Come funziona Astra Trident con NVE e NAE".
`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`.
`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
```

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.

Raccogliere i registri per la risoluzione dei problemi

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 di 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 stesso sull'host, i registri sono disponibili nell'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`
`storagePrefix`	Prefisso opzionale per i nomi dei volumi. Predefinito: `netappdvp_`.	`staging_`
`limitVolumeSize`	Restrizione opzionale sulle dimensioni dei volumi. Predefinito: "" (non applicato)	`10g`

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`	`10G`

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`	Indirizzo IP del protocollo LIF. **Driver NAS ONTAP**: Si consiglia di specificare `dataLIF`. Se non fornito, Astra Trident recupera i dati LIF dalla SVM. È possibile specificare un FQDN (Fully-qualified domain name) da utilizzare per le operazioni di montaggio NFS, consentendo di creare un DNS round-robin per il bilanciamento del carico tra più LIF di dati. **Driver SAN ONTAP**: Non specificare iSCSI. Astra Trident utilizza "Mappa LUN selettiva ONTAP" Per scoprire i LIF iSCI necessari per stabilire una sessione multi-percorso. Viene generato un avviso se `dataLIF` è esplicitamente definito.	`10.0.0.2`

Opzione	Descrizione	Esempio
`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`
`password`	Password per la connessione al dispositivo di storage	`secret`
`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`	Astra Trident crea e gestisce per nodo `igroups` come `netappdvp`. Questo valore non può essere modificato o omesso. **Disponibile solo per `ontap-san` driver**.	`netappdvp`
`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**.	`300g`

Opzione	Descrizione	Esempio
`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`
`sanType`	**Supportato per `ontap-san` solo conducente.** Utilizzare per selezionare `iscsi` Per iSCSI o. `nvme` Per NVMe/TCP.	`iscsi` se vuoto

Sono disponibili opzioni predefinite per evitare di doverle specificare su ogni volume creato:

Opzione	Descrizione	Esempio
`spaceReserve`	Modalità di prenotazione dello spazio; `none` (con thin provisioning) o. `volume` (spesso)	`none`
`snapshotPolicy`	Policy di Snapshot da utilizzare, l'impostazione predefinita è `none`	`none`
`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 a. `false`	`false`
`encryption`	Attiva NetApp Volume Encryption (NVE) sul nuovo volume; l'impostazione predefinita è `false`. NVE deve essere concesso in licenza e abilitato sul cluster per utilizzare questa opzione. Se NAE è attivato sul backend, tutti i volumi forniti in Astra Trident saranno abilitati per NAE. Per ulteriori informazioni, fare riferimento a: "Come funziona Astra Trident con NVE e NAE".	vero
`unixPermissions`	Opzione NAS per i volumi NFS con provisioning, per impostazione predefinita su `777`	`777`

Opzione	Descrizione	Esempio
`snapshotDir`	Opzione NAS per l'accesso a `.snapshot` directory, l'impostazione predefinita è `false`	`true`
`exportPolicy`	Opzione NAS per la policy di esportazione NFS da utilizzare, per impostazione predefinita a. `default`	`default`
`securityStyle`	Opzione NAS per l'accesso al volume NFS fornito. Supporto di NFS `mixed` e. `unix` stili di sicurezza. L'impostazione predefinita è `unix`.	`unix`
`fileSystemType`	OPZIONE SAN per selezionare il tipo di file system, l'impostazione predefinita è `ext4`	`xfs`
`tieringPolicy`	Policy di tiering da utilizzare, l'impostazione predefinita è `none`; `snapshot-only` Per la configurazione SVM-DR precedente a ONTAP 9.5	`none`

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.
- Le dimensioni minime consigliate per il membro/volume FlexGroup sono 100GiB.
- 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 <code>ontap-nas</code> driver

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1",
    "defaults": {
      "size": "10G",
      "spaceReserve": "none",
      "exportPolicy": "default"
    }
}
```

Esempio NFS per <code>ontap-nas-flexgroup</code> driver

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "password",
    "defaults": {
      "size": "100G",
      "spaceReserve": "none",
      "exportPolicy": "default"
    }
}
```

Esempio NFS per <code>ontap-nas-economy</code> driver

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1"
}
```

Esempio iSCSI per il <code>ontap-san</code> driver

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1",
    "igroupName": "netappdvp"
}
```

Esempio NFS per <code>ontap-san-economy</code> driver

```
{
    "version": 1,
    "storageDriverName": "ontap-san-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi_eco",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1",
    "igroupName": "netappdvp"
}
```

NVMe/TCP esempio per <code>ontap-san</code> driver

```
{
  "version": 1,
  "backendName": "NVMeBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nvme",
  "username":"vsadmin",
  "password":"password",
  "sanType": "nvme",
  "useREST": true
}
```

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`	<a href="https://<login>:<password>@<mvip>/json-rpc/<element-version&gt" class="bare">https://<login>:<password>@<mvip>/json-rpc/<element-version&gt;	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	`default`
`Types`	Specifiche QoS	Vedere l'esempio riportato di seguito

Opzione	Descrizione	Esempio
`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 volumi esistenti, sarà necessario impostare questo valore per accedere ai vecchi volumi 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
            }
        }
    ]
}
```

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.

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.

Panoramica della piattaforma

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. Fare riferimento a. "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.

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, consultare "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, fare riferimento a. "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 in grado di pianificare 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. 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. Fare riferimento a. "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 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

Selezionare e implementare un driver back-end per il sistema storage.

Driver backend ONTAP

I driver di back-end ONTAP si differenziano in base al protocollo utilizzato e al modo in cui i volumi vengono forniti nel sistema di storage. Pertanto, prendere in considerazione attentamente quando si decide quale driver 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).

I driver backend ONTAP disponibili sono:

- `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]	Sì	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 PV granulare
>
> Yes [3]: Non gestito da Astra Trident e non da PV granulare
>
> Yes [4]: Supportato per i volumi raw-block
>
> Yes [5]: Supportato da Astra 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.

Driver backend 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.

Driver backend Amazon FSX per ONTAP

Amazon FSX per NetApp ONTAP ti permette di sfruttare le caratteristiche, le performance e le capacità amministrative di NetApp che conosci bene, sfruttando al contempo la semplicità, l'agilità, la sicurezza e la scalabilità dello storage dei dati su AWS. FSX per ONTAP supporta molte funzioni di file system ONTAP e API di amministrazione. I driver compatibili per Cloud Volume ONTAP sono `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san` e. `ontap-san-economy`.

Driver backend 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 pagina:
>
> Yes [1]: Non gestito da Astra Trident
>
> Yes [2]: Supportato per i volumi raw-block

Driver backend 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 pagina:
>
> Yes [1]: Non gestito da Astra Trident

Driver backend Cloud Volumes Service su Google Cloud

Astra Trident utilizza `gcp-cvs` Driver per il collegamento a Cloud Volumes Service su Google Cloud.

Il `gcp-cvs` Il driver utilizza pool virtuali per astrarre il backend e consentire ad Astra Trident di determinare il posizionamento del volume. L'amministratore definisce i pool virtuali in `backend.json` file. Le classi di storage utilizzano selettori per identificare i pool virtuali in base all'etichetta.

- Se i pool virtuali sono definiti nel backend, Astra Trident tenterà di creare un volume nei pool di storage di Google Cloud a cui tali pool virtuali sono limitati.
- Se i pool virtuali non sono definiti nel backend, Astra Trident selezionerà un pool di storage Google Cloud dai pool di storage disponibili nella regione.

Per configurare il backend di Google Cloud su Astra Trident, è necessario specificare `projectNumber`, `apiRegion`, e. `apiKey` nel file backend. Il numero del progetto si trova nella console di Google Cloud. La chiave API viene presa dal file della chiave privata dell'account di servizio creato durante la configurazione dell'accesso API per Cloud Volumes Service su Google Cloud.

Per informazioni sui tipi di servizio e sui livelli di servizio di Cloud Volumes Service in Google Cloud, fare riferimento a. "Scopri di più sul supporto di Astra Trident per CVS per GCP".

Driver Cloud Volumes Service per Google Cloud	Snapshot	Cloni	Multi-attach	QoS	Espandere	Replica
`gcp-cvs`	Sì	Sì	Sì	Sì	Sì	Disponibile solo sul tipo di servizio CVS-Performance.

Note sulla replica

- La replica non è gestita da Astra Trident.
- Il clone verrà creato nello stesso pool di storage del volume di origine.

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.

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.

Emulare le policy di 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.

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.

Pool virtuali

Sono disponibili pool virtuali per tutti i backend Astra Trident. È possibile definire i pool virtuali per qualsiasi backend, utilizzando qualsiasi driver fornito da Astra Trident.

I pool 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 pool di storage, Astra Trident sceglierà una di queste da cui eseguire il provisioning del volume.

Progettazione di un pool 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 pool virtuali, questo problema è stato risolto. Virtual 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 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 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.

Emulazione di diversi livelli di servizio/QoS

È possibile progettare pool virtuali per l'emulazione delle 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 Azure NetApp Files con più etichette, che rappresentano diversi livelli di prestazioni. 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 virtuali. Utilizzando il `parameters.selector` Ciascun StorageClass richiama i pool virtuali che possono essere utilizzati per ospitare un volume.

Assegnazione di un insieme specifico di aspetti

È possibile progettare più pool virtuali con un set 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 virtuali. I volumi con cui viene eseguito il provisioning sul back-end avranno gli aspetti definiti nel pool virtuale 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, ciò 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 automaticamente 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. Aggiungerà automaticamente (e sovrascriverà se necessario) le dimensioni del volume del 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 della registrazione durante l'installazione iniziale di OpenShift e distribuzione della registrazione dopo l'installazione di OpenShift
installato.

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 ONTAP NFS non presenta questi problemi e può facilmente ripristinare una distribuzione di registrazione. 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 venga utilizzato lo storage corretto da parte dei servizi sottostanti. Esaminiamo le opzioni per ciascun metodo di distribuzione.

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.
`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 di 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 logging, il servizio di metriche può essere implementato durante una configurazione iniziale del cluster o dopo il suo funzionamento 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 stai eseguendo 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 per loro, 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 e disaster recovery

Scopri le opzioni di protezione e ripristino per Astra Trident e i volumi creati con Astra Trident. È necessario disporre di una strategia di protezione e ripristino dei dati per ogni applicazione con un requisito di persistenza.

Replica e recovery di Astra Trident

È possibile creare un backup per ripristinare Astra Trident in caso di disastro.

Replica di Astra Trident

Astra Trident utilizza i CRD Kubernetes per memorizzare e gestire il proprio stato e il cluster Kubernetes etcd per memorizzare i propri metadati.

Fasi

1. Eseguire il backup del cluster Kubernetes etcd utilizzando "Kubernetes: Backup di un cluster etcd".
2. Posizionare gli artefatti di backup su un FlexVol.

> Si consiglia di proteggere la SVM in cui risiede FlexVol con una relazione SnapMirror con un'altra SVM.

Recovery di Astra Trident

Utilizzando i CRD di Kubernetes e lo snapshot del cluster etcd di Kubernetes, puoi ripristinare Astra Trident.

Fasi

1. Dalla SVM di destinazione, montare il volume contenente i file di dati e i certificati Kubernetes etcd sull'host che verrà configurato come nodo master.
2. Copiare tutti i certificati richiesti relativi al cluster Kubernetes in `/etc/kubernetes/pki` e i file membri etcd sotto `/var/lib/etcd`.
3. Ripristinare il cluster Kubernetes dal backup etcd utilizzando "Kubernetes: Ripristino di un cluster etcd".
4. Eseguire `kubectl get crd` Per verificare che tutte le risorse personalizzate Trident siano state create e recuperare gli oggetti Trident per verificare che tutti i dati siano disponibili.

Replica e recovery di SVM

Astra Trident non può configurare le relazioni di replica, tuttavia l'amministratore dello storage può utilizzare "SnapMirror di ONTAP" Per replicare una SVM.

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.

A proposito di questa attività

Quando si utilizza la funzione di replica SVM di SnapMirror, considerare quanto segue:

- È necessario creare un backend distinto per ogni SVM con SVM-DR abilitato.
- Configurare le classi di storage in modo che selezionino i backend replicati solo quando necessario, per evitare volumi che non richiedono il provisioning della replica sui backend che supportano SVM-DR.
- Gli amministratori delle applicazioni devono comprendere i costi e la complessità aggiuntivi associati alla replica e considerare attentamente il piano di ripristino prima di iniziare questo processo.

Replica SVM

È possibile utilizzare "ONTAP: Replica SVM SnapMirror" Per creare la relazione di replica SVM.

SnapMirror consente di impostare le opzioni per il controllo degli elementi da replicare. È necessario conoscere le opzioni selezionate durante la preformatura Ripristino SVM con Astra Trident.

- "-identity-preserve true" Replica l'intera configurazione SVM.
- "-discard-configs network" Esclude le LIF e le relative impostazioni di rete.
- "-identity-preserve false" replica solo i volumi e la configurazione della sicurezza.

Ripristino SVM con Astra Trident

Astra Trident non rileva automaticamente gli errori SVM. In caso di disastro, l'amministratore può avviare manualmente il failover di Trident sulla nuova SVM.

Fasi

1. Annullare i trasferimenti SnapMirror pianificati e in corso, interrompere la relazione di replica, arrestare la SVM di origine e attivare la SVM di destinazione di SnapMirror.
2. Se specificato `-identity-preserve false` oppure `-discard-config network` Durante la configurazione della replica SVM, aggiornare `managementLIF` e. `dataLIF` Nel file di definizione backend Trident.
3. Confermare `storagePrefix` È presente nel file di definizione backend Trident. Questo parametro non può essere modificato. Omettere `storagePrefix` l'aggiornamento del backend non riesce.
4. Aggiornare tutti i backend richiesti per riflettere il nuovo nome SVM di destinazione utilizzando:

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n
<namespace>
```

5. Se specificato `-identity-preserve false` oppure `discard-config network`, è necessario eseguire il bounce di tutti i pod di applicazioni.

Se specificato `-identity-preserve true`, Tutti i volumi forniti da Astra Trident iniziano a servire i dati quando viene attivata la SVM di destinazione.

Replica e recovery dei volumi

Astra Trident non è in grado di configurare le relazioni di replica di SnapMirror, tuttavia l'amministratore dello storage può utilizzare "Replica e ripristino di ONTAP SnapMirror" Per replicare i volumi creati da Astra Trident.

È quindi possibile importare i volumi recuperati in Astra Trident utilizzando "importazione di volumi tridentctl".

L'importazione non è supportata su `ontap-nas-economy`, `ontap-san-economy`, o. `ontap-flexgroup-economy` driver.

Protezione dei dati Snapshot

È possibile proteggere e ripristinare i dati utilizzando:

- Un controller di snapshot esterno e CRD per creare snapshot di volumi Kubernetes di volumi persistenti (PVS).

  "Snapshot dei volumi"

- Snapshot ONTAP per ripristinare l'intero contenuto di un volume o per ripristinare singoli file o LUN.

  "Istantanee di ONTAP"

Replica dell'applicazione Astra Control Center

Grazie a Astra Control, è possibile replicare le modifiche di dati e applicazioni da un cluster all'altro utilizzando le funzionalità di replica asincrona di SnapMirror.

"Astra Control: Replica delle applicazioni su un sistema remoto utilizzando la tecnologia SnapMirror"

Sicurezza

Sicurezza

Utilizza i consigli elencati di seguito per assicurarti 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`.
Fare riferimento a. "" Per capire come Astra Trident configura CHAP sui backend ONTAP.

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. Una volta installato Astra Trident, gestisce i segreti CHAP e li memorizza in un `tridentvolume` Oggetto CR per il rispettivo PV. Quando si crea un PV, Astra Trident utilizza i segreti CHAP per avviare una sessione iSCSI e comunicare con il sistema NetApp HCI e SolidFire tramite CHAP.

I volumi creati da Astra Trident non sono associati ad alcun Gruppo di accesso ai volumi.

Utilizzare Astra Trident con NVE e NAE

NetApp ONTAP offre la crittografia dei dati inattivi per proteggere i dati sensibili in caso di furto, restituzione o riordinamento di un disco. Per ulteriori informazioni, fare riferimento a. "Panoramica sulla configurazione di NetApp Volume Encryption".

- Se NAE è attivato sul backend, qualsiasi volume fornito in Astra Trident sarà abilitato per NAE.
- Se NAE non è attivato sul backend, qualsiasi volume fornito in Astra Trident sarà abilitato per NVE, a meno che non si imposta il flag di crittografia NVE su `false` nella configurazione back-end.

I volumi creati in Astra Trident su un backend abilitato NAE devono essere crittografati con NVE o NAE.

- È possibile impostare il flag di crittografia NVE su `true` Nella configurazione backend Trident per eseguire l'override della crittografia NAE e utilizzare una chiave di crittografia specifica per volume.
- Impostazione del flag di crittografia NVE su `false` Su un backend abilitato NAE verrà creato un volume abilitato NAE. Non è possibile disattivare la crittografia NAE impostando il flag di crittografia NVE su `false`.

- È possibile creare manualmente un volume NVE in Astra Trident impostando esplicitamente il flag di crittografia NVE su `true`.

Per ulteriori informazioni sulle opzioni di configurazione del backend, fare riferimento a:

- "Opzioni di configurazione SAN ONTAP"
- "Opzioni di configurazione NAS ONTAP"

Linux Unified Key Setup (LUKS)

È possibile abilitare la configurazione delle chiavi unificate Linux (LUKS) per crittografare i volumi SAN ONTAP e SAN ONTAP SU Astra Trident. Astra Trident supporta la rotazione delle passphrase e l'espansione dei volumi con crittografia LUKS.

In Astra Trident, i volumi con crittografia LUKS utilizzano il cifrario aes-xts-plain64 e la modalità, come consigliato da "NIST".

Prima di iniziare

- Sui nodi di lavoro deve essere installata la crittografia 2.1 o superiore (ma inferiore a 3.0). Per ulteriori informazioni, visitare il sito "Gitlab: Crittsetup".
- Per motivi di performance, consigliamo ai nodi di lavoro di supportare Advanced Encryption Standard New Instructions (AES-NI). Per verificare il supporto AES-NI, eseguire il seguente comando:

```
grep "aes" /proc/cpuinfo
```

Se non viene restituito nulla, il processore non supporta AES-NI. Per ulteriori informazioni su AES-NI, visita: "Intel: Advanced Encryption Standard Instructions (AES-NI)".

Attivare la crittografia LUKS

È possibile attivare la crittografia lato host per volume utilizzando la configurazione unificata delle chiavi di Linux per volumi SAN ONTAP e SAN ONTAP.

Fasi

1. Definire gli attributi di crittografia LUKS nella configurazione del back-end. Per ulteriori informazioni sulle opzioni di configurazione back-end per ONTAP SAN, fare riferimento a. "Opzioni di configurazione SAN ONTAP".

```
"storage": [
    {
        "labels":{"luks": "true"},
        "zone":"us_east_1a",
        "defaults": {
            "luksEncryption": "true"
        }
    },
    {
        "labels":{"luks": "false"},
        "zone":"us_east_1a",
        "defaults": {
            "luksEncryption": "false"
        }
    },
]
```

2. Utilizzare `parameters.selector` Per definire i pool di storage utilizzando la crittografia LUKS. Ad esempio:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

3. Creare un segreto contenente la passphrase LUKS. Ad esempio:

```
kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA
```

Limitazioni

I volumi crittografati con LUKS non possono sfruttare la deduplica e la compressione ONTAP.

Configurazione back-end per l'importazione di volumi LUKS

Per importare un volume LUKS, è necessario impostare `luksEncryption` a.(`true` sul back-end. Il `luksEncryption` Option indica ad Astra Trident se il volume è conforme a LUKS (`true`) O non conforme a LUKS (`false`) come illustrato nell'esempio seguente.

```
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: admin
password: password
defaults:
  luksEncryption: 'true'
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

Ruotare una passphrase LUKS

È possibile ruotare la passphrase LUKS e confermare la rotazione.

> Non dimenticare una passphrase fino a quando non viene verificata la mancanza di riferimenti da qualsiasi volume, snapshot o segreto. In caso di perdita di una passphrase di riferimento, potrebbe non essere possibile montare il volume e i dati resteranno crittografati e inaccessibili.

A proposito di questa attività

La rotazione della passphrase LUKS si verifica quando viene creato un pod che monta il volume dopo aver specificato una nuova passphrase LUKS. Quando viene creato un nuovo pod, Astra Trident confronta la passphrase LUKS sul volume con la passphrase attiva nel segreto.

- Se la passphrase sul volume non corrisponde alla passphrase attiva nel segreto, si verifica la rotazione.
- Se la passphrase sul volume corrisponde alla passphrase attiva nel segreto, il `previous-luks-passphrase` il parametro viene ignorato.

Fasi

1. Aggiungere il `node-publish-secret-name` e. `node-publish-secret-namespace` Parametri StorageClass. Ad esempio:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-san
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/backendType: "ontap-san"
  csi.storage.k8s.io/node-stage-secret-name: luks
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-publish-secret-name: luks
  csi.storage.k8s.io/node-publish-secret-namespace: ${pvc.namespace}
```

2. Identificare le passphrase esistenti sul volume o sullo snapshot.

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["A"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["A"]
```

3. Aggiornare il segreto LUKS per il volume per specificare le passphrase nuove e precedenti. Assicurarsi `previous-luke-passphrase-name` e. `previous-luks-passphrase` associare la passphrase precedente.

```
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: B
  luks-passphrase: secretB
  previous-luks-passphrase-name: A
  previous-luks-passphrase: secretA
```

4. Creare un nuovo pod per il montaggio del volume. Questa operazione è necessaria per avviare la rotazione.
5. Verificare che la passphrase sia stata ruotata.

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["B"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["B"]
```

Risultati

La passphrase è stata ruotata quando viene restituita solo la nuova passphrase nel volume e nello snapshot.

> Se, ad esempio, vengono restituite due passphrase `luksPassphraseNames: ["B", "A"]`, la rotazione è incompleta. È possibile attivare un nuovo pod per tentare di completare la rotazione.

Abilitare l'espansione dei volumi

È possibile attivare l'espansione del volume su un volume crittografato con LUKS.

Fasi

1. Attivare il `CSINodeExpandSecret` feature gate (beta 1.25+). Fare riferimento a. "Kubernetes 1.25: Utilizza Secrets per l'espansione basata su nodi di volumi CSI" per ulteriori informazioni.
2. Aggiungere il `node-expand-secret-name` e. `node-expand-secret-namespace` Parametri StorageClass. Ad esempio:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-expand-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-expand-secret-namespace: ${pvc.namespace}
allowVolumeExpansion: true
```

Risultati

Quando si avvia l'espansione dello storage online, il kubelet passa le credenziali appropriate al driver.

Configurare la crittografia Kerberos in-flight

Con Astra Control Provivisioner, puoi migliorare la sicurezza dell'accesso ai dati abilitando la crittografia per il traffico tra il cluster gestito e il backend dello storage.

Astra Control Provisioner supporta la crittografia Kerberos su connessioni NFSv3 e NFSv4 da Red Hat

OpenShift e dai cluster Kubernetes upstream a volumi ONTAP on-premise.

Puoi creare, eliminare, ridimensionare, creare snapshot, clonare clone di sola lettura e importare i volumi che utilizzano la crittografia NFS.

Configura la crittografia Kerberos in-flight con i volumi ONTAP on-premise

Puoi attivare la crittografia Kerberos sul traffico storage tra il cluster gestito e un backend dello storage ONTAP on-premise.

> La crittografia Kerberos per il traffico NFS con backend di storage ONTAP on-premise è supportata solo utilizzando il `ontap-nas` driver di storage.

Prima di iniziare

- Assicurarsi di avere "Abilitato Astra Control Provisioner" sul cluster gestito.
- Assicurarsi di avere accesso all' `tridentctl` utilità.
- Assicurarsi di disporre dell'accesso come amministratore al back-end dello storage ONTAP.
- Conoscere il nome del volume o dei volumi che si desidera condividere dal back-end dello storage ONTAP.
- Verificare di aver preparato la VM di storage ONTAP per supportare la crittografia Kerberos per i volumi NFS. Fare riferimento alla "Attivare Kerberos su una LIF dati" per le istruzioni.
- Verificare che tutti i volumi NFSv4 utilizzati con la crittografia Kerberos siano configurati correttamente. Consultare la sezione Configurazione di dominio NetApp NFSv4 (pagina 13) della "Guida ai miglioramenti e alle Best practice di NetApp NFSv4".

Aggiungere o modificare criteri di esportazione ONTAP

Devi aggiungere regole alle policy di esportazione ONTAP esistenti o creare nuove policy di esportazione che supportino la crittografia Kerberos per il volume root delle macchine virtuali di storage ONTAP, oltre a qualsiasi volume ONTAP condiviso con il cluster Kubernetes upstream. Le regole dei criteri di esportazione aggiunte o i nuovi criteri di esportazione creati devono supportare i seguenti protocolli di accesso e autorizzazioni di accesso:

Protocolli di accesso

Configura la policy di esportazione con i protocolli di accesso NFS, NFSv3 e NFSv4.

Dettagli di accesso

È possibile configurare una delle tre diverse versioni della crittografia Kerberos, a seconda delle esigenze del volume:

- **Kerberos 5** - (autenticazione e crittografia)
- **Kerberos 5i** - (autenticazione e crittografia con protezione dell'identità)
- **Kerberos 5p** - (autenticazione e crittografia con protezione di identità e privacy)

Configurare la regola dei criteri di esportazione ONTAP con le autorizzazioni di accesso appropriate. Ad esempio, se i cluster montano i volumi NFS con una combinazione di crittografia Kerberos 5i e Kerberos 5p, utilizza le seguenti impostazioni di accesso:

Tipo	Accesso in sola lettura	Accesso in lettura/scrittura	Accesso superutente
UNIX	Attivato	Attivato	Attivato
Kerberos 5i	Attivato	Attivato	Attivato
Kerberos 5p	Attivato	Attivato	Attivato

Per informazioni su come creare policy di esportazione e regole delle policy di esportazione di ONTAP, consulta la seguente documentazione:

- "Creare una policy di esportazione"
- "Aggiungere una regola a un criterio di esportazione"

Creazione di un backend dello storage

Puoi creare una configurazione backend dello storage Astra Control Provivisioner che include funzionalità di crittografia Kerberos.

A proposito di questa attività

Quando si crea un file di configurazione backend di archiviazione che configura la crittografia Kerberos, è possibile specificare una delle tre diverse versioni della crittografia Kerberos utilizzando il `spec.nfsMountOptions` parametro:

- `spec.nfsMountOptions: sec=krb5` (autenticazione e crittografia)
- `spec.nfsMountOptions: sec=krb5i` (autenticazione e crittografia con protezione dell'identità)
- `spec.nfsMountOptions: sec=krb5p` (autenticazione e crittografia con protezione di identità e privacy)

Specificare un solo livello Kerberos. Se si specificano più livelli di crittografia Kerberos nell'elenco dei parametri, viene utilizzata solo la prima opzione.

Fasi

1. Nel cluster gestito, creare un file di configurazione backend dello storage utilizzando l'esempio seguente. Sostituire i valori tra parentesi <> con le informazioni dell'ambiente:

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-ontap-nas-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  storageDriverName: "ontap-nas"
  managementLIF: <STORAGE_VM_MGMT_LIF_IP_ADDRESS>
  dataLIF: <PROTOCOL_LIF_FQDN_OR_IP_ADDRESS>
  svm: <STORAGE_VM_NAME>
  username: <STORAGE_VM_USERNAME_CREDENTIAL>
  password: <STORAGE_VM_PASSWORD_CREDENTIAL>
  nasType: nfs
  nfsMountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
  qtreesPerFlexvol:
  credentials:
    name: backend-ontap-nas-secret
```

2. Utilizzare il file di configurazione creato nel passaggio precedente per creare il backend:

```
tridentctl create backend -f <backend-configuration-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.

Creare una classe di storage

È possibile creare una classe di archiviazione per il provisioning dei volumi con la crittografia Kerberos.

A proposito di questa attività

Quando si crea un oggetto classe di archiviazione, è possibile specificare una delle tre diverse versioni della crittografia Kerberos utilizzando il `mountOptions` parametro:

- `mountOptions: sec=krb5` (autenticazione e crittografia)
- `mountOptions: sec=krb5i` (autenticazione e crittografia con protezione dell'identità)
- `mountOptions: sec=krb5p` (autenticazione e crittografia con protezione di identità e privacy)

Specificare un solo livello Kerberos. Se si specificano più livelli di crittografia Kerberos nell'elenco dei parametri, viene utilizzata solo la prima opzione. Se il livello di crittografia specificato nella configurazione backend di archiviazione è diverso dal livello specificato nell'oggetto della classe di archiviazione, l'oggetto della classe di archiviazione ha la precedenza.

Fasi

1. Creare un oggetto Kubernetes StorageClass, usando il seguente esempio:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-sc
provisioner: csi.trident.netapp.io
mountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
parameters:
  backendType: "ontap-nas"
  storagePools: "ontapnas_pool"
  trident.netapp.io/nasType: "nfs"
allowVolumeExpansion: True
```

2. Creare la classe di storage:

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. Assicurarsi che la classe di archiviazione sia stata creata:

```
kubectl get sc ontap-nas-sc
```

   L'output dovrebbe essere simile a quanto segue:

```
NAME            PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

Provisioning dei volumi

Dopo aver creato un backend di storage e una classe di storage, è ora possibile eseguire il provisioning di un volume. Fare riferimento a queste istruzioni per "provisioning di un volume".

Configurare la crittografia Kerberos in-flight con i volumi Azure NetApp Files

È possibile attivare la crittografia Kerberos sul traffico di storage tra il cluster gestito e un singolo backend di storage Azure NetApp Files o un pool virtuale di backend di storage Azure NetApp Files.

Prima di iniziare

- Assicurati di aver abilitato Astra Control Provivisioner sul cluster Red Hat OpenShift gestito. Fare riferimento alla "Abilita Astra Control Provisioner" per le istruzioni.
- Assicurarsi di avere accesso all' `tridentctl` utilità.
- Assicurarsi di aver preparato il backend di archiviazione Azure NetApp Files per la crittografia Kerberos annotando i requisiti e seguendo le istruzioni riportate in "Documentazione Azure NetApp Files".
- Verificare che tutti i volumi NFSv4 utilizzati con la crittografia Kerberos siano configurati correttamente. Consultare la sezione Configurazione di dominio NetApp NFSv4 (pagina 13) della "Guida ai miglioramenti e alle Best practice di NetApp NFSv4".

Creazione di un backend dello storage

È possibile creare una configurazione backend dello storage Azure NetApp Files che include la funzionalità di crittografia Kerberos.

A proposito di questa attività

Quando si crea un file di configurazione backend dello storage che configura la crittografia Kerberos, è possibile definirlo in modo che venga applicato a uno dei due livelli possibili:

- Il **livello backend di archiviazione** utilizzando il `spec.kerberos` campo
- Il **livello pool virtuale** utilizzando il `spec.storage.kerberos` campo

Quando si definisce la configurazione a livello del pool virtuale, il pool viene selezionato utilizzando l'etichetta nella classe di archiviazione.

In entrambi i livelli, è possibile specificare una delle tre diverse versioni della crittografia Kerberos:

- `kerberos: sec=krb5` (autenticazione e crittografia)
- `kerberos: sec=krb5i` (autenticazione e crittografia con protezione dell'identità)
- `kerberos: sec=krb5p` (autenticazione e crittografia con protezione di identità e privacy)

Fasi

1. Nel cluster gestito, creare un file di configurazione backend dello storage utilizzando uno dei seguenti esempi, a seconda del punto in cui occorre definire il backend dello storage (livello di backend dello storage o livello del pool virtuale). Sostituire i valori tra parentesi <> con le informazioni dell'ambiente:

Esempio di livello di backend di archiviazione

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

Esempio di livello del pool virtuale

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  storage:
    - labels:
        type: encryption
      kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

2. Utilizzare il file di configurazione creato nel passaggio precedente per creare il backend:

```
tridentctl create backend -f <backend-configuration-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.

Creare una classe di storage

È possibile creare una classe di archiviazione per il provisioning dei volumi con la crittografia Kerberos.

Fasi

1. Creare un oggetto Kubernetes StorageClass, usando il seguente esempio:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "nfs"
  selector: "type=encryption"
```

2. Creare la classe di storage:

```
kubectl create -f sample-input/storage-class-sc-nfs.yaml
```

3. Assicurarsi che la classe di archiviazione sia stata creata:

```
kubectl get sc sc-nfs
```

L'output dovrebbe essere simile a quanto segue:

```
NAME          PROVISIONER             AGE
sc-nfs    csi.trident.netapp.io   15h
```

Provisioning dei volumi

Dopo aver creato un backend di storage e una classe di storage, è ora possibile eseguire il provisioning di un volume. Fare riferimento a queste istruzioni per "provisioning di un volume".

Conoscenza e supporto

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?

A partire dalla release 24,02, Astra Trident viene rilasciato ogni quattro mesi: Febbraio, giugno e ottobre.

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. Endpoint Prometheus che possono essere utilizzati per raccogliere informazioni sul funzionamento di Astra Trident, come il numero di backend gestiti, il numero di volumi con provisioning, i byte consumati e così via. È anche 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

Astra Trident supporta un'installazione offline da un registro privato?

Sì, Astra Trident può essere installato offline. Fare riferimento a. "Scopri di più sull'installazione di Astra Trident".

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 Trident crea un Kubernetes ClusterRole, che ha accesso specifico alle risorse PersistentVolume, PersistentVolumeClaim, StorageClass e Secret del cluster Kubernetes. Fare riferimento a. "Personalizzare l'installazione di tridentctl".

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. Fare riferimento a. "Personalizzare l'installazione di tridentctl".

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 "Astra 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 Astra "Discordare il canale" O GitHub.

La password del mio sistema storage è cambiata e Astra Trident non funziona più. Come posso eseguire il ripristino?

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 Trident vengono memorizzati in 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 è necessaria una correzione per i bug osservati dopo un aggiornamento, problemi di dipendenza o un aggiornamento non riuscito o incompleto, è necessario "Disinstallare Astra Trident" e reinstallare la versione precedente utilizzando le istruzioni specifiche per quella versione. Questo è l'unico modo consigliato per eseguire il downgrade a una versione precedente.

Gestione di back-end e volumi

È necessario definire le LIF di gestione e dati in un file di definizione back-end ONTAP?

La LIF di gestione è obbligatoria. La LIF dei dati varia:

- ONTAP SAN (SAN iSCSI): Non specificare iSCSI. Astra Trident utilizza "Mappa LUN selettiva ONTAP" Per scoprire i LIF iSCI necessari per stabilire una sessione multi-percorso. Viene generato un avviso se `dataLIF` è esplicitamente definito. Fare riferimento a. "Opzioni ed esempi di configurazione DELLA SAN ONTAP" per ulteriori informazioni.
- NAS ONTAP: Si consiglia di specificare `dataLIF`. Se non fornito, Astra Trident recupera i dati LIF dalla SVM. È possibile specificare un FQDN (Fully-qualified domain name) da utilizzare per le operazioni di montaggio NFS, consentendo di creare un DNS round-robin per il bilanciamento del carico tra più LIF di dati. Fare riferimento a. "Opzioni ed esempi di configurazione del NAS ONTAP" per ulteriori informazioni

Astra Trident può configurare CHAP per i backend ONTAP?

Sì. Astra Trident supporta il protocollo CHAP bidirezionale per i 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.

È possibile utilizzare gli indirizzi IPv6 per le LIF di gestione e dati?

Astra Trident supporta la definizione degli indirizzi IPv6 per:

- `managementLIF` e. `dataLIF` Per backend NAS ONTAP.
- `managementLIF` Per backend SAN ONTAP. Impossibile specificare `dataLIF` Su un backend SAN ONTAP.

Astra Trident deve essere installato utilizzando il flag `--use-ipv6` (per `tridentctl` installazione), `IPv6` (Per l'operatore Trident), o. `tridentTPv6` (Per l'installazione di Helm) per il suo funzionamento oltre 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?

È possibile aggiornare Data LIF su `ontap-nas` e. `ontap-nas-economy` solo.

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. Per ulteriori informazioni, fare riferimento a: "Come funziona Astra Trident con NVE e NAE"

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.

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.
```

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.

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.

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, fare riferimento a. "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 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 l' `tridentctl`

`uninstall` Comando per rimuovere Trident. Scaricare il desiderato "Versione di Trident" e installare utilizzando `tridentctl install` comando.

- 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.

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, devi dovrebbe dare un'occhiata a. `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:
    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. Dato che ogni cluster Kubernetes può solo farlo
Avere un'istanza di Trident, l'operatore garantisce che in qualsiasi dato caso
ora esiste un solo attivo `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 eseguire un'inizializzazione completa perché una o più immagini container non sono state recuperate.

Per risolvere il problema, modificare `TridentOrchestrator` CR.
In alternativa, è possibile eliminare `TridentOrchestrator`e creare un nuovo uno con la definizione modificata e precisa.

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.
```

Rimuovere completamente Astra Trident e i CRD

È possibile rimuovere completamente Astra Trident e tutti i CRD creati e le risorse personalizzate associate.

Questa operazione non può essere annullata. Non farlo a meno che non si desideri una nuova installazione di Astra Trident. Per disinstallare Astra Trident senza rimuovere i CRD, fare riferimento a. "Disinstallare Astra Trident".

Operatore Trident

Per disinstallare Astra Trident e rimuovere completamente i CRD utilizzando l'operatore Trident:

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

Timone

Per disinstallare Astra Trident e rimuovere completamente i CRD utilizzando Helm:

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

<code>tridentctl</code>

Per rimuovere completamente i CRD dopo aver disinstallato Astra Trident utilizzando `tridentctl`

```
tridentctl obliviate crd
```

Guasto durante l'unstadiazione del nodo NVMe con namespace di blocchi raw RWX o Kubernetes 1,26

Se utilizzi Kubernetes 1,26, il processo di staging del nodo potrebbe avere esito negativo quando utilizzi NVMe/TCP con namespace di blocchi raw RWX. I seguenti scenari forniscono una soluzione al problema. In alternativa, puoi eseguire l'upgrade di Kubernetes alla versione 1,27.

Eliminato il namespace e il pod

Prendi in considerazione uno scenario in cui hai un namespace gestito Astra Trident (volume persistente NVMe) collegato a un pod. Se si elimina lo spazio dei nomi direttamente dal back-end ONTAP, il processo di disinstallazione si blocca dopo aver tentato di eliminare il pod. Questo scenario non influisce sul cluster Kubernetes o su altre funzionalità.

Soluzione alternativa

Smontare il volume persistente (corrispondente a quel namespace) dal nodo rispettivo ed eliminarlo.

LIF dati bloccate

```
 If you block (or bring down) all the dataLIFs of the NVMe Astra Trident
backend, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
.Soluzione alternativa
Richiamare dataLIFS per ripristinare la funzionalità completa.
```

Mapping spazio dei nomi eliminato

```
 If you remove the `hostNQN` of the worker node from the corresponding
subsystem, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
.Soluzione alternativa
Aggiungere il `hostNQN` tornare al sottosistema.
```

Mapping spazio dei nomi eliminato

Quando si utilizza nvme/tcp in ambienti k8s con nomi di nodi lunghi, i pod non possono collegarsi allo storage.

Soluzione alternativa

Accorciare il nome del nodo in modo che il nome del sottosistema sia inferiore a 64 caratteri.

Supporto

NetApp offre supporto per Astra Trident in vari modi. Sono disponibili numerose opzioni di supporto self-service gratuite 24 ore su 24, 7 giorni su 7, come articoli della knowledge base (KB) e un canale di discording.

Ciclo di vita del supporto di Astra Trident

Astra Trident offre tre livelli di supporto basati sulla tua versione. Fare riferimento a. "Supporto delle versioni software NetApp per le definizioni".

Supporto completo
: Astra Trident offre il supporto completo per dodici mesi dalla data di rilascio.

Supporto limitato
: Astra Trident offre supporto limitato per i mesi compresi tra 13 e 24 dalla data di rilascio.

Supporto autonomo
: La documentazione di Astra Trident è disponibile per i mesi dal 25 al 36 dalla data di rilascio.

Versione	Supporto completo	Supporto limitato	Supporto autonomo
"24,02"	—	Febbraio 2026	Febbraio 2027

Versione	Supporto completo	Supporto limitato	Supporto autonomo
"23,10"	—	Ottobre 2025	Ottobre 2026
"23,07"	—	Luglio 2025	Luglio 2026
"23.04"	—	Aprile 2025	Aprile 2026
"23,01"	—	—	Gennaio 2026
"22,10"	—	—	Ottobre 2025
"22,07"	—	—	Luglio 2025
"22.04"	—	—	Aprile 2025

Supporto autonomo

Per un elenco completo degli articoli per la risoluzione dei problemi, fare riferimento a. "Knowledge base di NetApp (accesso richiesto)". È inoltre possibile trovare informazioni sulla risoluzione dei problemi relativi ad Astra "qui".

Sostegno della community

Esiste una vibrante community pubblica di utenti di container (inclusi gli sviluppatori Astra Trident) nella nostra Astra "Discordare il canale". Questo è un ottimo posto per porre domande generali sul progetto e discutere argomenti correlati con colleghi che condividono la stessa opinione.

Assistenza tecnica NetApp

Per assistenza con Astra Trident, creare un pacchetto di supporto utilizzando `tridentctl logs -a -n trident` e inviarla a. `NetApp Support <Getting Help>`.

Per ulteriori informazioni

- "Blog di Astra"
- "Blog di Astra Trident"
- "Hub Kubernetes"
- "NetApp.io"

Riferimento

Porte Astra Trident

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

Porte Astra Trident

Astra Trident comunica tramite le seguenti porte:

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

La porta della sonda liveness/Readiness può essere modificata durante l'installazione utilizzando `--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" Sono il modo più semplice per interagire con l'API REST di Astra Trident, puoi utilizzare l'endpoint REST direttamente se preferisci.

Quando utilizzare l'API REST

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

Per una maggiore sicurezza, Astra Trident `REST API` è limitato 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.

Utilizzo dell'API REST

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

L'API funziona come segue:

OTTIENI

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

Elenca tutti gli oggetti di quel tipo.

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

Ottiene i dettagli dell'oggetto denominato.

POST

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

Crea un oggetto del tipo specificato.

- Richiede una configurazione JSON per la creazione dell'oggetto. Per le specifiche di ciascun tipo di oggetto, fare riferimento alla "Gestisci Astra Trident usando tridentctl".
- Se l'oggetto esiste già, il comportamento varia: I backend aggiornano l'oggetto esistente, mentre tutti gli altri tipi di oggetto non riescono a eseguire l'operazione.

ELIMINARE

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

Elimina la risorsa denominata.

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

Opzioni della riga di comando

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

Registrazione

-debug

Attiva l'output di debug.

-loglevel <level>

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

Kubernetes

-k8s_pod

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.

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 un socket di dominio UNIX.

-config <file>

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

RIPOSO

-address <ip-or-host>

Specifica l'indirizzo in cui il server di GESTIONE DI Trident deve ascoltare. L'impostazione predefinita è localhost. Quando si ascolta su localhost e si esegue all'interno di un pod Kubernetes, l'interfaccia REST non è direttamente accessibile dall'esterno del pod. Utilizzare `-address ""` Per rendere 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 sulla quale il server di GESTIONE DI Trident deve ascoltare. Il valore predefinito è 8000.

-rest

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

Kubernetes e Trident Objects

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

In che modo gli oggetti interagiscono tra loro?

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

1. Un utente crea un `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, 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

Annotazione	Opzione volume	Driver supportati
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 dover utilizzare l'implementazione CSI.

Ecco un esempio: Se un utente ha già un PVC chiamato `mysql`, L'utente può creare un nuovo PVC chiamato `mysqlclone` utilizzando l'annotazione, 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. Fare riferimento a. Per una descrizione completa dei parametri e delle impostazioni associati 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/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters:
  <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

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

I parametri della classe di storage sono:

Attributo	Tipo	Obbligatorio	Descrizione
attributi	map[string]string	no	Vedere la sezione attributi riportata di seguito

Attributo	Tipo	Obbligatorio	Descrizione
StoragePools	map[string]StringList	no	Mappatura dei nomi backend negli elenchi di pool di storage all'interno di
AddtionalStoragePools	map[string]StringList	no	Mappa dei nomi backend a elenchi di pool di storage all'interno di
EsclusiveStoragePools	map[string]StringList	no	Mappa dei nomi backend in elenchi di pool di storage all'interno di

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

Attributi di selezione del pool di storage

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

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

Attributo	Tipo	Valori	Offerta	Richiesta	Supportato da
cloni	bool	vero, falso	Il pool supporta la clonazione dei volumi	Volume con cloni attivati	ontap-nas, ontap-san, solidfire-san, gcp-cvs
crittografia	bool	vero, falso	Il pool supporta volumi crittografati	Volume con crittografia attivata	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
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	Kubernetes Versione
Fstype	stringa	ext4, ext3, xfs, ecc.	Il tipo di file system per il blocco volumi	solidfire-san, ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy	Tutto
AllowVolumeExpansion	booleano	vero, falso	Abilitare o disabilitare il supporto per aumentare le dimensioni del PVC	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, gcp-cvs, azure-netapp-files	1.11+
VolumeBindingMode	stringa	Immediato, WaitForFirstConsumer	Scegliere quando si verifica il binding del volume e il provisioning dinamico	Tutto	1.19 - 1.26

- Il `fsType` 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. Fare riferimento a. "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.
- Fare riferimento a. "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/v1
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.

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.

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

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.

Astra Trident utilizza `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 disinstalla Trident utilizzando `tridentctl uninstall` Comando, i pod Trident vengono cancellati ma i CRD creati non vengono ripuliti. Fare riferimento a. "Disinstallare Trident" Per capire come Trident può essere completamente rimosso e riconfigurato da zero.

Astra Trident `StorageClass` oggetti

Trident crea classi di storage corrispondenti per Kubernetes `StorageClass` oggetti che specificano `csi.trident.netapp.io` 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 di backend Astra Trident

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

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

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

Astra 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.

Astra 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
dimensione	stringa	sì	Dimensione del volume per il provisioning in byte
protocollo	stringa	no	Tipo di protocollo da utilizzare; "file" o "blocco"

Attributo	Tipo	Obbligatorio	Descrizione
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 provisioning
processo.

Astra 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 backup
volume.

Astra Trident `ResourceQuota` oggetto

Il deambonset di Trident consuma un `system-node-critical` Classe di priorità - la classe di priorità più alta disponibile in Kubernetes - per garantire che Astra Trident sia in grado di identificare e pulire i volumi durante lo shutdown dei nodi aggraziati e consentire ai pod demonset di Trident di prevenire i carichi di lavoro con una priorità inferiore nei cluster in cui vi è un'elevata pressione sulle risorse.

Per ottenere questo risultato, Astra Trident impiega un `ResourceQuota` Scopo di garantire che una classe di priorità "system-node-critical" sul demonset Trident sia soddisfatta. Prima dell'implementazione e della creazione di demonset, Astra Trident cerca il `ResourceQuota` e, se non rilevato, lo applica.

Se è necessario un maggiore controllo sulla quota di risorse e sulla classe di priorità predefinite, è possibile generare un `custom.yaml` in alternativa, configurare `ResourceQuota` Oggetto che utilizza il grafico Helm.

Di seguito viene riportato un esempio di oggetto `ResourceQuota`che dà priorità al demonset Trident.

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
     matchExpressions:
       - operator : In
         scopeName: PriorityClass
         values: ["system-node-critical"]
```

Per ulteriori informazioni sulle quote delle risorse, fare riferimento a. "Kubernetes: Quote delle risorse".

Pulizia `ResourceQuota` se l'installazione non riesce

Nei rari casi in cui l'installazione non riesce dopo `ResourceQuota` l'oggetto viene creato, primo tentativo "disinstallazione in corso" quindi reinstallare.

In caso contrario, rimuovere manualmente `ResourceQuota` oggetto.

Rimuovere `ResourceQuota`

Se preferisci controllare la tua allocazione delle risorse, puoi rimuovere Astra Trident `ResourceQuota` oggetto mediante il comando:

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

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

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

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

Contesto di sicurezza Kubernetes obbligatorio e campi correlati

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

Funzionalità	Il `SYS_ADMIN` la funzionalità viene fornita come parte delle funzionalità predefinite per i container con privilegi. Ad esempio, Docker imposta queste funzionalità per i container con privilegi:
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

Seccomp	Il profilo Seccomp è sempre "non confinato" in container con privilegi; pertanto, non può essere abilitato in Astra Trident.
SELinux	In OpenShift, i container con privilegi vengono eseguiti in `spc_t` ("Super Privileged container") e i container senza privilegi vengono eseguiti in `container_t` dominio. Acceso `containerd`, con `container-selinux` installati, tutti i container vengono eseguiti in `spc_t` Domain, che disattiva effettivamente SELinux. Pertanto, Astra Trident non aggiunge `seLinuxOptions` ai container.
DAC	I container con privilegi devono essere eseguiti come root. I container non privilegiati vengono eseguiti come root per accedere ai socket unix richiesti da CSI.

Standard di sicurezza Pod (PSS)

Etichetta	Descrizione	Predefinito

`pod-security.kubernetes.io/enforce`
 	
`pod-security.kubernetes.io/enforce-version`
	Consente di ammettere il controller Trident e i nodi nello spazio dei nomi install.
	
	Non modificare l'etichetta dello spazio dei nomi.
		`enforce: privileged`
		
		`enforce-version: <version of the current cluster or highest version of PSS tested.>`

> La modifica delle etichette dello spazio dei nomi può causare la mancata pianificazione dei pod, un "errore di creazione: …" Oppure "Warning: trident-csi-…". In tal caso, controllare se l'etichetta dello spazio dei nomi di `privileged` è stato modificato. In tal caso, reinstallare Trident.

Policy di sicurezza Pod (PSP)

Campo	Descrizione	Predefinito
`allowPrivilegeEscalation`	I container con privilegi devono consentire l'escalation dei privilegi.	`true`
`allowedCSIDrivers`	Trident non utilizza volumi effimeri CSI inline.	Vuoto
`allowedCapabilities`	I container Trident non con privilegi non richiedono più funzionalità rispetto al set predefinito e ai container con privilegi vengono concesse tutte le funzionalità possibili.	Vuoto
`allowedFlexVolumes`	Trident non utilizza un "Driver FlexVolume", quindi non sono inclusi nell'elenco dei volumi consentiti.	Vuoto
`allowedHostPaths`	Il pod di nodi Trident monta il filesystem root del nodo, quindi non c'è alcun beneficio nell'impostazione di questo elenco.	Vuoto
`allowedProcMountTypes`	Trident non ne utilizza alcuno `ProcMountTypes`.	Vuoto
`allowedUnsafeSysctls`	Trident non richiede alcuna operazione non sicura `sysctls`.	Vuoto
`defaultAddCapabilities`	Non è necessario aggiungere funzionalità ai container con privilegi.	Vuoto
`defaultAllowPrivilegeEscalation`	L'escalation dei privilegi viene gestita in ogni pod Trident.	`false`
`forbiddenSysctls`	No `sysctls` sono consentiti.	Vuoto

Campo	Descrizione	Predefinito
`fsGroup`	I container Trident vengono eseguiti come root.	`RunAsAny`
`hostIPC`	Il montaggio dei volumi NFS richiede l'IPC host per comunicare con `nfsd`	`true`
`hostNetwork`	Iscsiadm richiede che la rete host comunichi con il daemon iSCSI.	`true`
`hostPID`	Per verificare se è necessario utilizzare il PID host `rpc-statd` è in esecuzione sul nodo.	`true`
`hostPorts`	Trident non utilizza porte host.	Vuoto
`privileged`	I pod di nodi Trident devono eseguire un container privilegiato per poter montare i volumi.	`true`
`readOnlyRootFilesystem`	I pod di nodi Trident devono scrivere nel file system del nodo.	`false`
`requiredDropCapabilities`	I pod di nodi Trident eseguono un container privilegiato e non possono rilasciare funzionalità.	`none`
`runAsGroup`	I container Trident vengono eseguiti come root.	`RunAsAny`
`runAsUser`	I container Trident vengono eseguiti come root.	`runAsAny`
`runtimeClass`	Trident non utilizza `RuntimeClasses`.	Vuoto
`seLinux`	Trident non viene impostato `seLinuxOptions` Perché ci sono attualmente differenze nel modo in cui i runtime dei container e le distribuzioni Kubernetes gestiscono SELinux.	Vuoto
`supplementalGroups`	I container Trident vengono eseguiti come root.	`RunAsAny`
`volumes`	I pod Trident richiedono questi plug-in di volume.	`hostPath, projected, emptyDir`

SCC (Security Context Constraints)

Etichette	Descrizione	Predefinito
`allowHostDirVolumePlugin`	I pod di nodi Trident montano il filesystem root del nodo.	`true`

Etichette	Descrizione	Predefinito
`allowHostIPC`	Il montaggio dei volumi NFS richiede l'IPC host per comunicare con `nfsd`.	`true`
`allowHostNetwork`	Iscsiadm richiede che la rete host comunichi con il daemon iSCSI.	`true`
`allowHostPID`	Per verificare se è necessario utilizzare il PID host `rpc-statd` è in esecuzione sul nodo.	`true`
`allowHostPorts`	Trident non utilizza porte host.	`false`
`allowPrivilegeEscalation`	I container con privilegi devono consentire l'escalation dei privilegi.	`true`
`allowPrivilegedContainer`	I pod di nodi Trident devono eseguire un container privilegiato per poter montare i volumi.	`true`
`allowedUnsafeSysctls`	Trident non richiede alcuna operazione non sicura `sysctls`.	`none`
`allowedCapabilities`	I container Trident non con privilegi non richiedono più funzionalità rispetto al set predefinito e ai container con privilegi vengono concesse tutte le funzionalità possibili.	Vuoto
`defaultAddCapabilities`	Non è necessario aggiungere funzionalità ai container con privilegi.	Vuoto
`fsGroup`	I container Trident vengono eseguiti come root.	`RunAsAny`
`groups`	Questo SCC è specifico di Trident ed è vincolato al proprio utente.	Vuoto
`readOnlyRootFilesystem`	I pod di nodi Trident devono scrivere nel file system del nodo.	`false`
`requiredDropCapabilities`	I pod di nodi Trident eseguono un container privilegiato e non possono rilasciare funzionalità.	`none`
`runAsUser`	I container Trident vengono eseguiti come root.	`RunAsAny`
`seLinuxContext`	Trident non viene impostato `seLinuxOptions` Perché ci sono attualmente differenze nel modo in cui i runtime dei container e le distribuzioni Kubernetes gestiscono SELinux.	Vuoto

Etichette	Descrizione	Predefinito
`seccompProfiles`	I container privilegiati vengono sempre eseguiti "senza confinare".	Vuoto
`supplementalGroups`	I container Trident vengono eseguiti come root.	`RunAsAny`
`users`	Viene fornita una voce per associare SCC all'utente Trident nello spazio dei nomi Trident.	n/a.
`volumes`	I pod Trident richiedono questi plug-in di volume.	`hostPath, downwardAPI, projected, emptyDir`

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/.

Informazioni sul copyright

Copyright © 2026 NetApp, Inc. Tutti i diritti riservati. Stampato negli Stati Uniti d'America. Nessuna porzione di questo documento soggetta a copyright può essere riprodotta in qualsiasi formato o mezzo (grafico, elettronico o meccanico, inclusi fotocopie, registrazione, nastri o storage in un sistema elettronico) senza previo consenso scritto da parte del detentore del copyright.

Il software derivato dal materiale sottoposto a copyright di NetApp è soggetto alla seguente licenza e dichiarazione di non responsabilità:

IL PRESENTE SOFTWARE VIENE FORNITO DA NETAPP "COSÌ COM'È" E SENZA QUALSIVOGLIA TIPO DI GARANZIA IMPLICITA O ESPRESSA FRA CUI, A TITOLO ESEMPLIFICATIVO E NON ESAUSTIVO, GARANZIE IMPLICITE DI COMMERCIABILITÀ E IDONEITÀ PER UNO SCOPO SPECIFICO, CHE VENGONO DECLINATE DAL PRESENTE DOCUMENTO. NETAPP NON VERRÀ CONSIDERATA RESPONSABILE IN ALCUN CASO PER QUALSIVOGLIA DANNO DIRETTO, INDIRETTO, ACCIDENTALE, SPECIALE, ESEMPLARE E CONSEQUENZIALE (COMPRESI, A TITOLO ESEMPLIFICATIVO E NON ESAUSTIVO, PROCUREMENT O SOSTITUZIONE DI MERCI O SERVIZI, IMPOSSIBILITÀ DI UTILIZZO O PERDITA DI DATI O PROFITTI OPPURE INTERRUZIONE DELL'ATTIVITÀ AZIENDALE) CAUSATO IN QUALSIVOGLIA MODO O IN RELAZIONE A QUALUNQUE TEORIA DI RESPONSABILITÀ, SIA ESSA CONTRATTUALE, RIGOROSA O DOVUTA A INSOLVENZA (COMPRESA LA NEGLIGENZA O ALTRO) INSORTA IN QUALSIASI MODO ATTRAVERSO L'UTILIZZO DEL PRESENTE SOFTWARE ANCHE IN PRESENZA DI UN PREAVVISO CIRCA L'EVENTUALITÀ DI QUESTO TIPO DI DANNI.

NetApp si riserva il diritto di modificare in qualsiasi momento qualunque prodotto descritto nel presente documento senza fornire alcun preavviso. NetApp non si assume alcuna responsabilità circa l'utilizzo dei prodotti o materiali descritti nel presente documento, con l'eccezione di quanto concordato espressamente e per iscritto da NetApp. L'utilizzo o l'acquisto del presente prodotto non comporta il rilascio di una licenza nell'ambito di un qualche diritto di brevetto, marchio commerciale o altro diritto di proprietà intellettuale di NetApp.

Il prodotto descritto in questa guida può essere protetto da uno o più brevetti degli Stati Uniti, esteri o in attesa di approvazione.

LEGENDA PER I DIRITTI SOTTOPOSTI A LIMITAZIONE: l'utilizzo, la duplicazione o la divulgazione da parte degli enti governativi sono soggetti alle limitazioni indicate nel sottoparagrafo (b)(3) della clausola Rights in Technical Data and Computer Software del DFARS 252.227-7013 (FEB 2014) e FAR 52.227-19 (DIC 2007).

I dati contenuti nel presente documento riguardano un articolo commerciale (secondo la definizione data in FAR 2.101) e sono di proprietà di NetApp, Inc. Tutti i dati tecnici e il software NetApp forniti secondo i termini del presente Contratto sono articoli aventi natura commerciale, sviluppati con finanziamenti esclusivamente privati. Il governo statunitense ha una licenza irrevocabile limitata, non esclusiva, non trasferibile, non cedibile, mondiale, per l'utilizzo dei Dati esclusivamente in connessione con e a supporto di un contratto governativo statunitense in base al quale i Dati sono distribuiti. Con la sola esclusione di quanto indicato nel presente documento, i Dati non possono essere utilizzati, divulgati, riprodotti, modificati, visualizzati o mostrati senza la previa approvazione scritta di NetApp, Inc. I diritti di licenza del governo degli Stati Uniti per il Dipartimento della Difesa sono limitati ai diritti identificati nella clausola DFARS 252.227-7015(b) (FEB 2014).

Informazioni sul marchio commerciale

NETAPP, il logo NETAPP e i marchi elencati alla pagina http://www.netapp.com/TM sono marchi di NetApp, Inc. Gli altri nomi di aziende e prodotti potrebbero essere marchi dei rispettivi proprietari.