Documentazione Trident 26.10

Trident

NetApp
November 10, 2026

This PDF was generated from https://docs.netapp.com/it-it/trident/index.html on November 10, 2026.
Always check docs.netapp.com for the latest.

Sommario

Documentazione Trident 26.10 . . . 1
Note di rilascio . . . 2
Novità . . . 2
Novità della versione 26.10.1 . . . 2
Novità della versione 26.10 . . . 2
Novità della versione 26.06.2 . . . 6
Novità della versione 26.06.1 . . . 7
Novità della versione 26.06 . . . 7
Novità della versione 26.02.1 . . . 10
Novità della versione 26.02 . . . 11
Novità della versione 25.10 . . . 14
Modifiche nel 25.06.2 . . . 16
Modifiche nel 25.06.1 . . . 16
Modifiche nel 25.06 . . . 16
Modifiche nel 25.02.1 . . . 19
Modifiche nel 25,02 . . . 19
Modifiche nel 24.10.1 . . . 21
Modifiche nel 24,10 . . . 21
Modifiche nel 24,06 . . . 23
Modifiche nel 24,02 . . . 24
Modifiche nel 23,10 . . . 24
Modifiche nel 23.07.1 . . . 25
Modifiche nel 23,07 . . . 25
Modifiche nel 23,04 . . . 26
Cambiamenti nel 23.01.1 . . . 27
Cambiamenti nel 23.01 . . . 27
Cambiamenti nel 22.10 . . . 28
Cambiamenti nel 22.07 . . . 29
Cambiamenti nel 22.04 . . . 30
Cambiamenti nel 22.01.1 . . . 31
Cambiamenti nel 22.01.0 . . . 31
Cambiamenti nel 21.10.1 . . . 32
Cambiamenti nel 21.10.0 . . . 32
Problemi noti . . . 33
Trova ulteriori informazioni . . . 34
Versioni precedenti della documentazione . . . 34
NetApp Trident supporto per sistemi di storage ONTAP ASA r2 . . . 35
Operazioni supportate . . . 35
Operazioni non supportate . . . 35
Problemi noti . . . 35
I volumi rimangono collegati dopo l’aggiornamento da Trident 22.10 o versioni precedenti . . . 36
VolumeSnapshots non raggiungono lo stato ReadyToUse . . . 36
Il ripristino dei backup di file di grandi dimensioni può non riuscire . . . 36

Inizia subito . . . 37
Scopri Trident . . . 37
Scopri Trident . . . 37
Architettura Trident . . . 38
Concetti . . . 41
Avvio rapido di Trident . . . 45
Quali sono le prossime novità? . . . 46
Requisiti Trident . . . 46
Informazioni critiche su Trident . . . 46
Frontend supportati (orchestratori) . . . 46
Back-end supportati (storage) . . . 47
Supporto Trident per KubeVirt e OpenShift Virtualization . . . 47
Supporto Trident per Kubernetes su Kubernetes e piani di controllo ospitati OpenShift . . . 48
Requisiti delle funzionalità . . . 49
Sistemi operativi host testati . . . 49
Configurazione dell'host . . . 49
Configurazione del sistema storage . . . 50
Porte Trident . . . 50
Immagini container e corrispondenti versioni di Kubernetes . . . 50
Installare Trident . . . 52
Informazioni sull'installazione di Trident . . . 52
Informazioni critiche su Trident 26.10 . . . 52
Prima di iniziare . . . 52
Scegliere il metodo di installazione desiderato . . . 52
Scegliere la modalità di installazione . . . 54
Selezionare il processo in base al metodo e alla modalità . . . 55
Passaggio da un metodo di installazione all'altro . . . 55
Altre opzioni di configurazione note . . . 55
Installare utilizzando l'operatore Trident . . . 56
Implementare manualmente l'operatore Trident (modalità standard) . . . 56
Implementare manualmente l'operatore Trident (modalità offline) . . . 61
Implementare l'operatore Trident utilizzando Helm (modalità standard) . . . 66
Implementare l'operatore Trident utilizzando Helm (modalità offline) . . . 75
Personalizzare l'installazione dell'operatore Trident . . . 84
Installare usando tridentctl . . . 95
Installare Trident usando tridentctl . . . 95
Personalizza l'installazione di Trident tridentctl . . . 99
Installare utilizzando un operatore certificato OpenShift . . . 100
Installare Trident utilizzando OpenShift OperatorHub . . . 100
Passa all'operatore Trident certificato OpenShift . . . 103
USA Trident . . . 107
Prepara il nodo worker di Trident . . . 107
Selezionare gli strumenti giusti . . . 107
Rilevamento del servizio del nodo . . . 107
Volumi NFS . . . 108

Volumi iSCSI . . . 108
Volumi NVMe/TCP . . . 112
SCSI su volumi FC . . . 113
Preparatevi al provisioning dei volumi SMB . . . 114
Esegui il provisioning dello storage a oggetti con COSI . . . 116
Panoramica . . . 116
Requisiti e limitazioni . . . 117
Abilita COSI . . . 117
Configurare un archivio di oggetti StorageGRID . . . 117
Configura un archivio di oggetti ONTAP . . . 118
Crea una BucketClass e richiedi un bucket . . . 119
Concedi a un workload l'accesso a un bucket . . . 120
Configurare e gestire i backend . . . 121
Configurare i backend di Trident . . . 121
Azure NetApp Files . . . 121
Google Cloud NetApp Volumes . . . 138
Configurare un backend NetApp HCI o SolidFire . . . 175
Driver SAN ONTAP . . . 180
Driver NAS ONTAP . . . 209
Usa un posizionamento bilanciato con i backend ONTAP . . . 243
Amazon FSX per NetApp ONTAP . . . 245
Crea backend con kubectl . . . 279
Gestire i backend . . . 285
Creare e gestire classi di archiviazione . . . 295
Crea una classe di storage Trident . . . 295
Gestisci le classi di storage di Trident . . . 297
Provisioning e gestione dei volumi . . . 299
Esegui il provisioning di un volume Trident . . . 299
Sposta un volume . . . 302
Espandi i volumi Trident . . . 305
Comprendere i limiti del sottosistema RWX NVMe . . . 315
Scalabilità del controller . . . 316
Espansione automatica del volume . . . 321
Configura il dimensionamento automatico del volume ONTAP . . . 328
Gestisci le policy di Autogrow . . . 332
Importa i volumi Trident . . . 341
Personalizza i nomi e le etichette dei volumi di Trident . . . 352
Condividi un volume NFS Trident tra namespace . . . 355
Clona i volumi Trident tra namespace . . . 358
Replica dei volumi con SnapMirror . . . 361
Utilizzare la topologia CSI . . . 367
Lavora con gli snapshot di Trident . . . 374
Lavora con gli snapshot dei gruppi di volumi di Trident . . . 382
Gestire e monitorare Trident . . . 386
Upgrade Trident (Aggiorna server) . . . 386

Upgrade Trident (Aggiorna server) . . . 386
Eseguire l'upgrade con l'operatore . . . 387
Aggiorna Trident con tridentctl . . . 392
Gestisci Trident usando tridentctl . . . 392
Comandi e flag globali . . . 392
Opzioni di comando e flag . . . 394
Supporto plugin . . . 398
Monitora le operazioni di storage di Trident . . . 399
Panoramica . . . 399
Fase 1: Definire un target Prometheus . . . 399
Fase 2: Creazione di un ServiceMonitor Prometheus . . . 399
Fase 3: Eseguire una query sulle metriche di Trident con PromQL . . . 401
Ulteriori informazioni sulla telemetria di Trident AutoSupport . . . 402
Disattiva metriche Trident . . . 403
Disinstallare Trident . . . 403
Determinare il metodo di installazione originale . . . 404
Disinstallare un'installazione dell'operatore Trident . . . 404
Disinstallare un `tridentctl` installazione . . . 405
Trident per Docker . . . 406
Prerequisiti per l'implementazione . . . 406
Verificare i requisiti . . . 406
Strumenti NVMe . . . 408
Strumenti FC . . . 409
Implementa Trident . . . 410
Metodo del plugin gestito da Docker (versione 1.13/17.03 e successive) . . . 410
Metodo tradizionale (versione 1.12 o precedente) . . . 412
Aggiornare o disinstallare Trident . . . 415
Eseguire l'upgrade . . . 415
Disinstallare . . . 416
Lavorare con i volumi . . . 417
Creare un volume . . . 417
Rimuovere un volume . . . 418
Clonare un volume . . . 418
Accesso ai volumi creati esternamente . . . 419
Opzioni di volume specifiche del driver . . . 419
Raccogliere i log . . . 424
Raccogliere i registri per la risoluzione dei problemi . . . 424
Suggerimenti generali per la risoluzione dei problemi . . . 425
Gestione di più istanze di Trident . . . 425
Procedura per il plug-in gestito da Docker (versione 1.13/17.03 o successiva) . . . 425
Procedura per la versione tradizionale (1.12 o precedente) . . . 426
Opzioni di configurazione dello storage . . . 426
Opzioni di configurazione globale . . . 426
Configurazione di ONTAP . . . 427
Configurazione del software Element . . . 435

Problemi noti e limitazioni . . . 436
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. . . . 437
I nomi dei volumi devono contenere almeno 2 caratteri. . . . 437
Docker Swarm presenta comportamenti che impediscono a Trident di supportarlo con ogni combinazione di storage e driver. . . . 437
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. . . . 437
Best practice e consigli . . . 438
Raccomandazioni per l'implementazione di Trident . . . 438
Eseguire l'implementazione in uno spazio dei nomi dedicato . . . 438
Utilizza quote e limiti di intervallo per controllare il consumo dello storage . . . 438
Procedure consigliate per la configurazione dello storage di Trident . . . 438
Panoramica della piattaforma . . . 438
Best practice per ONTAP e Cloud Volumes ONTAP . . . 438
Best practice di SolidFire . . . 443
Dove trovare ulteriori informazioni? . . . 444
Integra Trident con Kubernetes . . . 445
Selezione e implementazione dei driver . . . 445
Design di classe storage . . . 448
Progettazione di un pool virtuale . . . 449
Operazioni di volume . . . 450
Servizio di metriche . . . 453
Protezione dei dati e disaster recovery di Trident . . . 455
Replica e recovery di Trident . . . 455
Replica e recovery di SVM . . . 455
Replica e recovery dei volumi . . . 456
Protezione dei dati Snapshot . . . 457
Automazione del failover delle applicazioni stateful con Trident . . . 457
Dettagli sulla forza di distacco . . . 457
Dettagli sul failover automatico . . . 458
Sicurezza . . . 462
Raccomandazioni di sicurezza di Trident . . . 463
Trident Linux Unified Key Setup (LUKS) . . . 464
Risolvere un Lease bloccato per i volumi RWX Linux Unified Key Setup (LUKS) . . . 470
Crittografia in transito Kerberos di Trident . . . 472
Proteggi le applicazioni con Trident Protect . . . 481
Scopri di più su Trident Protect . . . 481
Quali sono le prossime novità? . . . 481
Installa Trident Protect . . . 481
Requisiti Trident Protect . . . 481
Installa e configura Trident Protect . . . 485
Installa il plugin Trident Protect CLI . . . 488
Personalizza l'installazione Trident Protect . . . 491

Gestisci Trident Protect . . . 496
Gestisci l'autorizzazione e il controllo degli accessi Trident Protect . . . 496
Monitorare le risorse Trident Protect . . . 502
Genera un pacchetto di supporto Trident Protect . . . 506
Aggiorna Trident Protect . . . 508
Gestisci e proteggi le applicazioni . . . 509
Utilizzare gli oggetti Trident Protect AppVault per gestire i bucket . . . 510
Crittografa i metadati di AppVault per snapshot e backup di Trident Protect . . . 521
Definisci un'applicazione per la gestione con Trident Protect . . . 527
Proteggi le applicazioni utilizzando Trident Protect . . . 532
Utilizza gli snapshot dei gruppi di volumi con Trident Protect . . . 542
Ripristino delle applicazioni . . . 547
Replica delle applicazioni con NetApp SnapMirror . . . 570
Migrare le applicazioni utilizzando Trident Protect . . . 586
Proteggi le macchine virtuali KubeVirt con Trident Protect . . . 590
Gestire gli hook di esecuzione Trident Protect . . . 601
Disinstallare Trident Protect . . . 609
Blog Trident e Trident Protect . . . 610
Blog Trident . . . 610
Blog Trident Protect . . . 610
Conoscenza e supporto . . . 612
Domande frequenti . . . 612
Domande generali . . . 612
Installare e utilizzare Trident su un cluster Kubernetes . . . 612
Risoluzione dei problemi e supporto . . . 613
Upgrade Trident (Aggiorna server) . . . 614
Gestione di back-end e volumi . . . 615
Risoluzione dei problemi . . . 619
Risoluzione dei problemi generali . . . 619
Implementazione Trident non riuscita utilizzando l'operatore . . . 620
Implementazione Trident non riuscita utilizzando `tridentctl` . . . 622
Rimuovere completamente Trident e CRD . . . 622
Guasto durante l'unstadiazione del nodo NVMe con namespace di blocchi raw RWX o Kubernetes 1,26 . . . 623
I client NFSv4.2 segnalano "argomento non valido" dopo l'aggiornamento ONTAP quando si prevede che "v4.2-xattrs" sia abilitato . . . 624
Supporto . . . 624
Ciclo di vita del supporto Trident . . . 624
Supporto autonomo . . . 625
Sostegno della community . . . 625
Assistenza tecnica NetApp . . . 625
Per ulteriori informazioni . . . 625
Riferimento . . . 626
Porte Trident . . . 626
Panoramica . . . 626

API REST Trident . . . 628
Quando utilizzare l'API REST . . . 628
Utilizzo dell'API REST . . . 628
Opzioni della riga di comando . . . 629
Registrazione . . . 629
Kubernetes . . . 629
Docker . . . 630
RIPOSO . . . 630
Kubernetes e Trident Objects . . . 630
In che modo gli oggetti interagiscono tra loro? . . . 630
Kubernetes `PersistentVolumeClaim` oggetti . . . 631
Kubernetes `PersistentVolume` oggetti . . . 633
Kubernetes `StorageClass` oggetti . . . 633
Kubernetes `VolumeSnapshotClass` oggetti . . . 637
Kubernetes `VolumeSnapshot` oggetti . . . 637
Kubernetes `VolumeSnapshotContent` oggetti . . . 637
Oggetti Kubernetes `VolumeGroupSnapshotClass` . . . 638
Oggetti Kubernetes `VolumeGroupSnapshot` . . . 638
Oggetti Kubernetes `VolumeGroupSnapshotContent` . . . 639
Kubernetes `CustomResourceDefinition` oggetti . . . 639
OggettiTrident `StorageClass` . . . 639
Oggetti backend Trident . . . 640
OggettiTrident `StoragePool` . . . 640
OggettiTrident `Volume` . . . 640
OggettiTrident `Snapshot` . . . 642
OggettoTrident `ResourceQuota` . . . 642
Pod Security Standards (PSS) e Security Context Constraints (SCC) . . . 643
Contesto di sicurezza Kubernetes obbligatorio e campi correlati . . . 644
Standard di sicurezza Pod (PSS) . . . 645
Policy di sicurezza Pod (PSP) . . . 645
SCC (Security Context Constraints) . . . 646
Note legali . . . 649
Copyright . . . 649
Marchi . . . 649
Brevetti . . . 649
Direttiva sulla privacy . . . 649
Open source . . . 649

Documentazione Trident 26.10

Note di rilascio

Novità

Le note di rilascio forniscono informazioni sulle nuove funzionalità, sui miglioramenti e sulle correzioni di bug nell'ultima versione di NetApp 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 della versione 26.10.1

Scopri le novità in NetApp Trident e Trident Protect, inclusi miglioramenti, correzioni e deprecazioni.

Trident

Correzioni

- **Kubernetes**:
  - Risolto il problema per cui i pod del nodo Trident scollegavano forzatamente i volumi in uso all'avvio dopo l'aggiornamento da Trident 25.10 o versioni precedenti, causando l'errore dei pod con `volume … is not staged`.

Novità della versione 26.10

Scopri le novità in NetApp Trident e Trident Protect, inclusi miglioramenti, correzioni e deprecazioni.

> Non aggiornare a Trident 26.10.0. Durante l'avvio, i pod del nodo Trident potrebbero scollegare volumi ancora in uso. I pod interessati potrebbero non funzionare con l'errore `volume … is not staged`. Questo problema è risolto in Trident 26.10.1. Aggiorna invece a Trident 26.10.1 o versione successiva.

Trident

Miglioramenti

- **Kubernetes**:
  - Aggiunto il supporto per il provisioning dinamico basato su COSI dei bucket di storage a oggetti da StorageGRID. Per ulteriori informazioni, vedi "Esegui il provisioning dello storage a oggetti con COSI".
  - Aggiunto il supporto per l'aggiornamento periodico dei LIF dati NVMe per rilevare e adattarsi alle modifiche di configurazione. Per ulteriori informazioni, consulta "Aggiorna i LIF dei dati NVMe".
  - Aggiunto il supporto per il dimensionamento automatico di ONTAP sui nuovi volumi creati dai driver ONTAP-SAN e ONTAP-SAN-Economy. Per ulteriori informazioni, consulta "Configura il dimensionamento automatico del volume ONTAP".
  - Aggiunto il supporto per Kubernetes 1.37. Per le versioni supportate e le immagini richieste, consulta "Requisiti Trident".
  - Aggiunto il riconoscimento del PVC del clone di backup per i backup di CommVault e Rubrik per saltare

la ONTAP Volume Recovery Queue (problema "#1182").

- Aggiunta l' `maxNamespacesPerSuperSubsystem`opzione backend per i nuovi volumi NVMe RWX. Per ulteriori informazioni, consulta "Limiti del sottosistema NVMe RWX".
- Aggiunta la registrazione dei nodi CSI basata su CRD tramite `TridentNode` spec e status, in sostituzione del backchannel HTTPS del controller per le implementazioni Kubernetes.
- Aggiunto il supporto del programma di installazione per la proiezione dei token dell'account di servizio con un'audience configurabile per Google Cloud Workload Identity Federation (WIF) su OpenShift. Per i dettagli sull'installazione e la configurazione del backend, vedi "Configura Google Cloud NetApp Volumes con Workload Identity Federation".
- Aggiunta un' `backend_state`etichetta alla `trident_backend_info` metrica per identificare lo stato di salute dei singoli backend in Prometheus (Issue "#1088"). Per informazioni sulle metriche di Trident, vedi "Monitor Trident".

- Aggiunto il supporto per la crittografia LUKS per i volumi RWX sottoposti a provisioning dal driver ONTAP-SAN tramite iSCSI. Per ulteriori informazioni, consulta "Utilizzo di Linux Unified Key Setup (LUKS)".
- Aggiunto il supporto GA della concorrenza per il driver ONTAP-NAS-FlexGroup e Docker; la concorrenza di ASA r2 è ora GA, non più in Technical Preview. Per i driver supportati e il relativo comportamento, vedi "Scalabilità del controller".
- Aggiunto il supporto per l'espansione del volume CSI quando non tutti i percorsi iSCSI sono disponibili, utilizzando la rimozione del percorso e l'autoriparazione iSCSI. Per ulteriori informazioni, vedi "Espandere i volumi".
- Aggiunta l'annotazione `trident.netapp.io/unixGroupID` PVC per impostare, al momento della creazione, l'ID del gruppo UNIX del volume ONTAP per i volumi ONTAP-NAS e ONTAP-NAS-FlexGroup con root squash (Problema "#1079").
- Aggiunto il supporto di Talos OS per NVMe/TCP, NFSv4 e iSCSI (Issue "#806"). Per le piattaforme supportate e i requisiti dei nodi worker, vedi "Requisiti Trident" e "Preparare il nodo di lavoro".
- Aggiunto il parametro opzionale `storageAPITimeout` backend per configurare i timeout dei client ONTAP REST e ZAPI per ciascun backend.
- Aggiunta l' `useRESTSQLBypass`opzione di backend ONTAP per aggiungere `SQL=false` alla ricerca dei nomi REST di ONTAP.
- Promossa l'importazione qtree ONTAP-NAS-Economy da Technical Preview a GA. Per le istruzioni di importazione, vedi "Importa i volumi Trident".

Correzioni

- **Kubernetes**:
  - Corretto il problema per cui le regole della policy di esportazione ONTAP-NAS venivano eliminate erroneamente durante gli aggiornamenti del backend o i riavvii del controller quando `enableConcurrency` è `true` (Problemi "#1166", "#1179"). Per la configurazione della policy di esportazione, vedi "Opzioni ed esempi di configurazione del NAS ONTAP".
  - Corretti gli aggiornamenti non sostanziali del backend, come le modifiche alle credenziali o alla QoS, che attivavano scansioni degli orfani per volume e potevano bloccare le operazioni CSI sui backend di grandi dimensioni. Ora le scansioni si eseguono solo quando cambiano la SVM o il prefisso di storage.
  - Corrette le credenziali visualizzate nei log di debug, inclusi i dump delle richieste CSI gRPC, l'output dell'ambiente di avvio e `TridentBackendConfig`le specifiche registrate. L'`output di avvio ora registra i valori solo per le variabili Trident `TRIDENT_*` ed elenca le altre variabili per nome.

- Risolta una race condition tra la pubblicazione CSI e lo spostamento del volume che poteva collegare un volume con informazioni di accesso obsolete. Per i dettagli sullo spostamento del volume, consulta "Sposta un volume".
- Risolto `NodeGetInfo` il problema della restituzione di etichette di topologia vuote prima del completamento della registrazione del nodo, che poteva causare errori di provisioning basati sulla topologia (Problema "#1131"). Per il provisioning basato sulla topologia, consulta "Esegui il provisioning dello storage usando la topologia CSI".
- Risolta una rara race condition nella disconnessione NVMe che poteva causare l'arresto di un sottosistema condiviso mentre un altro pod si stava collegando.
- Risolti gli errori di pubblicazione dei cloni PVC NVMe e la mappatura errata del namespace padre durante la clonazione di un volume montato in modalità concorrente.
- Risolti gli errori di creazione degli snapshot per i PVC sui backend ONTAP-SAN-Economy configurati con `nameTemplate`. Per esempi di configurazione del backend, vedi "Opzioni ed esempi di configurazione DELLA SAN ONTAP".
- Risolto il problema per cui i backend REST di ONTAP lasciavano orfano un FlexVol appena creato se veniva eliminato prima che l'indice dei nomi REST si aggiornasse; Trident ora elimina i volumi tramite UUID.
- Risolto il problema del blocco all'avvio del controller quando un LIF di backend non è raggiungibile. Il backend interessato viene contrassegnato `Failed` e l'avvio prosegue.
- Risolto il problema per cui i pod del controller e del nodo gestiti dall'operatore si riavviavano a ogni riconciliazione quando configurati con un selettore di nodi multi-chiave. Per la configurazione del selettore di nodi, vedere "Personalizzare l'installazione dell'operatore Trident".
- Risolto il problema delle risorse Azure NetApp Files `TridentBackendConfig` create da TridentConfigurator a cui mancava l'etichetta `trident.netapp.io/configurator` richiesta dalla convalida post-creazione.
- Risolto il problema di registrazione del nodo che si verificava quando il servizio Trident CSI utilizzava un ClusterIP IPv6 (Problema "#1164").
- Risolto un problema per cui un nodo Trident veniva rimosso definitivamente quando un nodo worker veniva eliminato e si ricollegava mentre le pubblicazioni di volumi obsolete erano ancora in fase di annullamento della pubblicazione.
- Risolto il problema per cui i volumi ONTAP-NAS perdevano l'accesso all'esportazione NFS sui nodi già collegati quando la policy di esportazione memorizzata nella cache di Trident non era aggiornata. Per la configurazione della policy di esportazione, vedi "Opzioni ed esempi di configurazione del NAS ONTAP".
- Risolta `tridentctl` la pulizia dell'installazione che elimina il CRD del modello di correzione del nodo, il che potrebbe impedire installazioni successive con distacco forzato abilitato. Per i dettagli sull'installazione, vedi "Installare Trident usando tridentctl".
- Risolto il problema delle chiamate di eliminazione degli snapshot che sovraccaricavano l'API ONTAP aggiungendo un ritardo per i tentativi. Per le operazioni sugli snapshot, vedere "Gestisci le istantanee".
- Corretto il commento dell'helper dell'immagine del grafico Helm dell'operatore Trident AutoSupport e la virgolettatura del valore cloudProvider.
- Corretta la gestione di ONTAP ZAPI volume-create per processi occupati o esistenti, in modo che i tentativi di CSI non distruggano prematuramente FlexVols né causino cicli di creazione/distruzione.
- Risolto un panic durante il nuovo tentativo di eseguire snapshot di gruppi di volumi per un gruppo di coerenza che non esiste più. Per gli snapshot di gruppi di volumi, vedi "Creare snapshot del gruppo di volumi".

    - Rimosso l'avviso di supporto di Trident in base alle versioni minime e massime di Kubernetes supportate.
- Risolta una race condition nei lock condivisi con nome che poteva consentire operazioni sovrapposte sulla stessa risorsa.
- Risolto il problema di inizializzazione del driver ASA che si verificava quando non era possibile trovare uno snapshot.
- Risolto il provisioning GCNV per i pool regionali Flex Unified quando i nodi del cluster non si trovano nella zona HA primaria; Trident non considera più tale zona HA come una topologia CSI obbligatoria. I pool zonali rimangono invariati. Per la configurazione di Flex Unified, vedi "Configurare storage a blocchi (iSCSI)".
- Risolto il problema per cui i volumi GCNV Flex Unified NAS e SAN venivano erroneamente contrassegnati come orfani durante la riconciliazione del backend quando l'identità canonica del backend differiva dal nome del volume Trident (Problema "#1189"). Per la configurazione del backend GCNV, vedi "Configurare Google Cloud NetApp Volumes".
- Risolto il problema di rilevamento di NVMe/TCP sugli host in cui il trasporto è integrato nel kernel (Problema "#1019"). Per i requisiti dei nodi, vedere "Preparare il nodo di lavoro".
- Risolto il problema per cui ONTAP-NAS e ONTAP-NAS-FlexGroup ritentavano l'espansione all'infinito quando il volume ONTAP era già sufficientemente grande per la richiesta, inclusa la riserva per gli snapshot. Per l'espansione del volume, vedi "Espandere i volumi".
- Risolto il problema del montaggio di volumi fissi che non riusciva a causa di percorsi di giunzione mancanti. Per la configurazione NAS di ONTAP, vedi "Configura i driver NAS di ONTAP".
- Risolto il problema di creazione del volume ONTAP-SAN-Economy che non riusciva quando `storagePrefix` conteneva trattini. Per le opzioni di configurazione del backend, vedere "Opzioni ed esempi di configurazione DELLA SAN ONTAP".
- Corretta la pubblicazione del clone ONTAP-NAS di sola lettura con policy di esportazione automatiche, in modo da preservare le regole di esportazione del FlexVol di origine. Per la configurazione di ONTAP NAS, vedi "Opzioni ed esempi di configurazione del NAS ONTAP".
- Corretto il comportamento di nuovo tentativo del trasporto HTTP (Issue "#1139").
- Risolto il problema per cui il rilevamento dei dispositivi fantasma ignorava la rimozione del dispositivo SCSI (problema "#1161").
- Corretto il provisioning con gestione della concorrenza che selezionava una zona topologica diversa dalla zona preferita del nodo selezionato. Per informazioni sul comportamento in termini di concorrenza e topologia, vedi "Scalabilità del controller".
- Risolto il problema del posizionamento asimmetrico dei volumi tra backend equivalenti quando la concorrenza è abilitata; i backend vengono ora selezionati in modo casuale e uniforme, come nel provisioning non concorrente. Per i dettagli, consulta "Scalabilità del controller".
- Risolto il problema della registrazione di nodi con indirizzi IP mancanti, che poteva rimuovere le regole della policy di esportazione NFS quando la tabella degli indirizzi di rete dell'host cambiava durante l'avvio del pod del nodo Trident.
- Risolto il problema per cui la creazione del clone continuava oltre il polling REST e associava il PVC come FlexClone non suddiviso; Trident ora rileva i successivi errori del job di creazione.
- Risolto il problema relativo alla segnalazione di falsi errori di ridimensionamento durante l'espansione del volume ONTAP-SAN sul nodo. Per informazioni sull'espansione del volume, vedere "Espandere i volumi".

Miglioramenti sperimentali

Non utilizzare in ambienti di produzione.

- **[Anteprima tecnica] Kubernetes**: Aggiunto il supporto per il provisioning dinamico basato su COSI dei bucket di storage a oggetti da ONTAP S3. Per ulteriori informazioni, vedi "Esegui il provisioning dello storage a oggetti con COSI".

Trident Protect

Miglioramenti

- Aggiunto il supporto per la crittografia dei metadati di snapshot e backup di AppVault. Usa una password del vault personalizzata per ogni AppVault. La rotazione della password del vault crittografa di nuovo la chiave di crittografia dei metadati e aggiorna ogni repository Kopia e Restic nell'AppVault. Per saperne di più, consulta "Crittografa i metadati di AppVault per snapshot e backup di Trident Protect".
- Aggiunto il supporto per mantenere l'indirizzo MAC delle VM KubeVirt durante il ripristino, utile quando la licenza di un'applicazione è associata a un indirizzo MAC. Per abilitarlo, imposta l'annotazione protect.trident.netapp.io/retain-mac-address su "true". Per saperne di più, consulta "Utilizza le impostazioni di ripristino avanzate Trident Protect".
- Aggiunto il supporto per gli snapshot dei gruppi di volumi, che acquisiscono le richieste di volume persistente correlate di un'applicazione nello stesso momento. Definisci i gruppi di volumi in `spec.volumeGroups` nella risorsa personalizzata Application per etichetta, per macchina virtuale KubeVirt o entrambe. Per saperne di più, consulta "Utilizza gli snapshot dei gruppi di volumi con Trident Protect".
- Aggiunto il supporto per il failover a uno snapshot precedente di un'applicazione replicata, che ti consente di eseguire il ripristino da un punto nel tempo precedente al danneggiamento o alla compromissione dei dati. Per ulteriori informazioni, consulta "Esegui il failover a uno snapshot precedente usando Trident Protect".

Correzioni

- Risolto il problema per cui le installazioni di Trident Protect non riuscivano nel `trident-protect-pre-upgrade-hook` job perché mancava il `trident-protect-controller-manager` ServiceAccount ("Issue #1154").
- Corretti i ripristini solo PVC per includere un `ownerReference` solo quando il proprietario esiste.

Novità della versione 26.06.2

Scopri le novità in NetApp Trident e Trident Protect, inclusi miglioramenti, correzioni e deprecazioni.

Trident

Miglioramenti

Correzioni

- **Kubernetes**:
  - Risolto il problema per cui le regole della policy di esportazione di ONTAP-NAS venivano eliminate silenziosamente quando il controller Trident veniva riavviato o quando veniva aggiornato un `TridentBackendConfig` mentre `enableConcurrency` era impostato su `true`. Questo poteva lasciare bloccati i volumi già montati ("Issue #1166", "Issue #1179").

- Risolto il problema relativo agli aggiornamenti del backend che eseguivano una scansione degli orfani per volume alla ricerca di modifiche non sostanziali, come credenziali e QoS, che potevano bloccare le operazioni CSI per ore su backend di grandi dimensioni. La scansione ora viene eseguita solo quando cambia il prefisso SVM o di storage.
- Corrette le credenziali che comparivano nei log di debug di Trident, inclusi i dump delle richieste CSI gRPC, il dump dell'ambiente di avvio e `TridentBackendConfig` le specifiche registrate dal controller CRD. Il dump dell'ambiente di avvio ora registra i valori solo per le variabili di Trident `TRIDENT_*` ed elenca le altre variabili per nome.

Novità della versione 26.06.1

Scopri le novità in NetApp Trident e Trident Protect, inclusi miglioramenti, correzioni e deprecazioni.

Trident

Correzioni

- **Kubernetes**:
  - Risolto un problema di competizione tra la pubblicazione CSI e lo spostamento del volume che poteva collegare un volume utilizzando informazioni di accesso obsolete.
  - Risolto un problema di competizione in cui NodeGetInfo poteva restituire etichette di topologia vuote prima del completamento della registrazione del nodo, causando errori di provisioning basati sulla topologia.
  - Risolto un raro problema di condizione di gara nella disconnessione NVMe che poteva causare la chiusura di un sottosistema condiviso mentre un altro pod era ancora in fase di connessione.
  - Risolti gli errori di pubblicazione dei cloni NVMe PVC (e la mappatura errata del namespace padre) durante la creazione di cloni di un volume montato in modalità concorrente.
  - Risolti gli errori di creazione degli snapshot per i PVC sui backend ONTAP-SAN-Economy utilizzando un nameTemplate.

Miglioramenti

- **Kubernetes**:
  - Aggiunta l'opzione `storageAPITimeout` (secondi) nella configurazione del backend ONTAP per impostare i timeout dei client API REST e ZAPI per ciascun backend.

Novità della versione 26.06

Scopri le novità in NetApp Trident e Trident Protect, inclusi miglioramenti, correzioni e deprecazioni.

Trident

Miglioramenti

- **Kubernetes**:
  - Aggiunto il supporto generale (GA) per la concorrenza per i driver ONTAP-NAS-Economy e ONTAP-SAN-Economy, facendo uscire questa funzionalità dalla Tech Preview, e aggiunto il supporto GA per la concorrenza anche per i driver Azure NetApp Files (ANF) e SolidFire.
  - Aggiunto il supporto per Kubernetes 1.36.

- Aggiunto il supporto generale (GA) per l'autogrow dei volumi con i driver NAS e SAN di Google Cloud NetApp Volumes (GCNV).
- Aggiunto il posizionamento bilanciato Unified ONTAP per una selezione ottimale degli aggregati durante il provisioning di nuovi volumi ("Issue #64").
- Aggiunto il supporto ONTAP AFX per il driver ONTAP-NAS-Flexgroup.
- Aggiunta l'integrazione con il NetApp Shift Toolkit per semplificare la migrazione delle macchine virtuali quando usi il Red Hat OpenShift Migration Toolkit for Virtualization.
- Aggiunto il supporto Cloud Identity per il driver SAN GCNV.
- Aggiunto il supporto per i volumi subordinati per la propagazione di Trident Volume Publication (TVP) e i flussi di lavoro di crescita automatica in Trident.
- Aggiunto il supporto per gli aggiornamenti dei volumi e gli snapshot dei gruppi di volumi quando usi Trident con la concorrenza abilitata.
- Aggiunta la limitazione dinamica della frequenza gRPC per GCNV.
- Aggiunto un probe di esecuzione per la verifica della prontezza del controller e ottimizzate le impostazioni predefinite di QPS e Burst dell'API Kubernetes per il controller e i sidecar.
- Sequenziamento delle operazioni sui nodi migliorato grazie alla separazione delle fasi di collegamento, configurazione della crittografia e formattazione del filesystem.
- I filesystem root dei container Trident sono stati montati in modalità di sola lettura per migliorare la sicurezza.
- Aggiunta la rimozione best-effort degli snapshot dei volumi clone nei backend ONTAP quando viene eliminato lo snapshot di origine di un volume clone.
- Aggiunto il riconoscimento dei PVC di clonazione di backup per impedire che i PVC temporanei creati da Veeam Kasten, Trilio e Cohesity rimangano bloccati nella ONTAP Volume Recovery Queue.
- Gestione avanzata degli ARN AWS per il supporto di regioni speciali.
- Impedisci la riduzione delle dimensioni dei Flexvol durante l'operazione di ridimensionamento per il driver ONTAP-SAN-ECONOMY.

Miglioramenti sperimentali

Non utilizzare in ambienti di produzione.

- **[Anteprima tecnica]**: Aggiunto il supporto per l'importazione di qtree ONTAP-NAS-Economy.
- **[Anteprima tecnica]**: Aggiunto il supporto per la concorrenza per i driver ASA-R2 (SAN e NVMe).
- **[Anteprima tecnica]**: Aggiunto trasporto HTTP GCNV in modalità ONTAP per i driver ONTAP-SAN e ONTAP-NAS.

Correzioni

- **Kubernetes**:
  - Risolti i problemi di race condition e la gestione dello stato per le operazioni simultanee di pubblicazione/annullamento pubblicazione, clonazione, cache e aggiornamento del backend.
  - Risolte le condizioni di competizione delle policy di esportazione e la gestione simultanea di pubblicazione/annullamento pubblicazione per i volumi subordinati e i cloni di sola lettura nei driver ONTAP-NAS e ONTAP-NAS-Economy.
  - Corretto il rilevamento ONTAP-NAS-Economy FlexVol dopo un failover MetroCluster quando le policy

di snapshot differiscono ("Issue #1082").

- Risolti i tentativi di eliminazione del volume ONTAP-NAS-Economy che interrompevano le eliminazioni di qtree di lunga durata ("Issue #1121").
- Risolto il comportamento di importazione e ridimensionamento di ONTAP-SAN e ONTAP-SAN-Economy, inclusa la convalida di fsType, la gestione dei metadati del volume e il comportamento della modalità di autogrow.
- Risolto un problema di race condition nello spazio dei nomi NVMe/TCP durante la creazione simultanea di volumi ("Issue #1089").
- Risolti gli errori di ridimensionamento iSCSI che si verificavano quando i percorsi e i portali disponibili non corrispondevano e migliorato il comportamento dei ridimensionamenti ripetuti.
- Risolti gli errori di eliminazione dei volumi ONTAP-SAN causati da condizioni di volume occupato.
- Corretto la gestione degli errori ONTAP REST e migliorato il comportamento del timeout dello stato SVM.
- Gestione corretta del pool zonale SAN GCNV e del gruppo host, conversione normalizzata dell'identificativo del volume e normalizzazione delle richieste di creazione al di sotto della dimensione minima.
- Corretta la selezione del pool di capacità nativo GCNV per escludere i pool in modalità ONTAP.
- Risolto il failover di KubeVirt per le VM che utilizzano Trident Automated Workload Failover per velocizzare il failover delle VM.
- Risolti i problemi RBAC delle installazioni gestite dall'operatore, inclusi i ClusterRoles dei nodi mancanti e la riconciliazione errata di ClusterRole e ClusterRoleBinding durante gli aggiornamenti. Questo vale per le installazioni basate su Operator e Helm che utilizzano il chart Trident Operator.
- Risolti diff persistenti di GitOps su TridentConfigurator e CRD correlati omettendo i valori di priorità predefiniti e correggendo l'indentazione del template YAML ("Issue #915", "Issue #1120").
- Corretto il patching dei CRD TridentVolumePublication durante un upgrade di Helm ("Issue #1151").
- Corretto l'ordine di avvio di CSI avviando il servizio CSI gRPC prima dei tentativi di registrazione del nodo.
- Risolta la sovrapposizione della configurazione di clonazione SMB sicura durante le operazioni simultanee.
- Corrette le annotazioni mancanti sul pod controller Trident dopo un aggiornamento ("Issue #1004").
- Provisioning di qtree lenti migliorato su SVM di grandi dimensioni ("Issue #1142").
- Corretto ControllerPublish per usare il tipo di filesystem di configurazione del volume invece di un tipo di filesystem predefinito errato.
- Risolto un problema per cui i dispositivi dm orfani potevano causare ripetuti errori CSI NodeUnstageVolume.
- Pulizia migliorata dei nodi obsoleti e dei percorsi multipli, rimuovendo gli artefatti fantasma dei percorsi multipli e le risorse TridentNode obsolete dopo la pulizia dei nodi eliminati in modo soft.

Trident Protect

Miglioramenti

- Aggiunta l'integrazione con OpenShift Service CA Operator per automatizzare la gestione dei certificati TLS per il webhook di ammissione di Trident Protect. Abilita impostando `useOpenshiftServiceCA` su `true` nei tuoi valori Helm. Per saperne di più, consulta "Personalizza l'installazione Trident Protect".

- Aggiunto il supporto per il backup e il ripristino delle VM KubeVirt su OpenShift Virtualization. Proteggi tutte le VM in un namespace oppure seleziona VM specifiche per nome e ripristina singole VM da un backup o da uno snapshot. Per saperne di più, consulta "Proteggi le macchine virtuali KubeVirt con Trident Protect".
- Aggiunto il supporto per le trasformazioni delle risorse, che ti permette di aggiungere, rimuovere e modificare gli attributi delle risorse Kubernetes durante il processo di ripristino. Per saperne di più, consulta "Modifica le risorse durante il ripristino utilizzando risorse personalizzate".
- Aggiunto il supporto per `storageClassMapping` in AppMirrorRelationship (AMR), che consente la mappatura delle classi di archiviazione per i singoli PVC all'interno di una singola definizione AMR. Per saperne di più, consulta "Configura e gestisci la replica delle applicazioni utilizzando Trident Protect".

Correzioni

- Risolto il problema per cui il ripristino di macchine virtuali tra cluster non riusciva a causa di un errore del webhook di ammissione quando lo spazio dei nomi dell'applicazione di origine non esisteva nel cluster di destinazione.
- Risolto il problema per cui i ripristini selettivi in-place delle macchine virtuali non riuscivano nella fase di creazione dell'applicazione a causa di una mancata corrispondenza delle specifiche con una risorsa personalizzata dell'applicazione esistente; ora l'applicazione esistente viene riutilizzata.
- Risolto il problema per cui i processi di rispristino in-place delle macchine virtuali si bloccavano in uno stato `Running` quando sia la macchina virtuale che il suo PersistentVolumeClaim venivano eliminati; un PersistentVolumeClaim mancante viene ora considerato come un errore irreversibile.
- Risolto il problema che causava l'attivazione errata di un backup o di uno snapshot immediato durante la modifica di una pianificazione di protezione esistente.
- Corretto il finalizzatore dello snapshot che tentava di recuperare la risorsa personalizzata dell'applicazione dopo che era stata eliminata; ora il finalizzatore salta quando l'applicazione non viene trovata.
- La capacità di archiviazione protetta fissa aumenta significativamente dopo un ripristino in-place di una macchina virtuale grazie all'utilizzo della copia sparsa durante la fase di copia di PersistentVolumeClaim.
- Risolto il problema per cui Trident Protect eliminava l'oggetto snapshot di Kubernetes prima di confermare l'eliminazione dello snapshot sottostante in ONTAP; ora il finalizzatore attende l'eliminazione di `VolumeSnapshot`.
- Risolto il problema per cui gli snapshot non venivano eseguiti quando un'operazione di freeze di una macchina virtuale non riusciva; ora le operazioni di freeze e unfreeze continuano anche in caso di errore.
- Risolto il problema per cui le applicazioni delle macchine virtuali KubeVirt ricreavano in modo errato i PersistentVolumeClaims durante le operazioni di reverse, reverse resync e resync di AppMirrorRelationship.

Novità della versione 26.02.1

Scopri le novità in NetApp Trident e Trident Protect, inclusi miglioramenti, correzioni e deprecazioni.

Trident

Correzioni

- **Kubernetes**:
- Corretto il comportamento SAN di Google Cloud NetApp Volumes (GCNV) per supportare i pool Flex Unified ZONAL per carichi di lavoro a blocchi (iSCSI). Sono ora supportate sia le configurazioni zonali che regionali.

- Corretta la gestione dei pool NAS UNIFIED di Google Cloud NetApp Volumes (GCNV) migliorando la creazione, la convalida, la denominazione e le operazioni di ricerca dei volumi di fallback quando `InternalID` è assente.
- Risolto un problema per cui le operazioni di ridimensionamento dei dispositivi iSCSI potevano non riuscire quando i percorsi e i portali disponibili non corrispondevano.
- Risolto un problema intermittente che `blkid` identificava in modo errato il tipo di filesystem per i dispositivi LUKS, impedendo la creazione di nuovi volumi.
- È stato risolto un problema per cui le operazioni di pubblicazione e staging di LUKS su larga scala potevano bloccare il controller Trident. Gli utenti sono ora tenuti a tenere traccia delle passphrase LUKS associate ai volumi.
- Risolto un problema per cui le partizioni iSCSI multipath potevano apparire come dispositivi fantasma, impedendo le operazioni CSI `NodeUnstage` di avere successo.
- Risolto un problema di race condition nello spazio dei nomi NVMe/TCP durante la creazione simultanea di volumi (Issue #1089).
- Risolto il problema delle operazioni di clonazione simultanee degli snapshot quando questi venivano eseguiti nello stesso secondo.
- Risolto un problema che impediva alle VM KubeVirt che utilizzano `dataVolumes` di eseguire il failover con Trident Automated Workload Failover.
- Risolto un errore che si verificava durante l'eliminazione di una risorsa personalizzata non valida `TridentMirrorRelationship` mentre si utilizzava il core concorrente di Trident.
- Risolto il problema di indentazione nella `tridentactionmirrorupdates` definizione YAML del CRD di Kubernetes (Issue #1120).
- Risolto il problema relativo alla chiamata API REST per l'eliminazione asincrona di qtree (Issue #1121).
- Corretto `ControllerPublish` per utilizzare il tipo di file system di configurazione del volume anziché un tipo di file system predefinito errato.
- Risolto il problema di registrazione all'avvio del controller per segnalare correttamente lo stato dei driver concorrenti.
- Gestione corretta dei backend sospesi in Trident concorrente.
- Gestione avanzata degli ARN AWS per il supporto di regioni speciali.

Trident Protect

Correzioni

Aggiornate le dipendenze di Trident Protect per risolvere le vulnerabilità di sicurezza:

- gRPC-Go: "CVE-2026-33186"
- Go x/crypto SSH: "CVE-2024-45337"
- OpenTelemetry-Go: "CVE-2026-24051"
- Runtime Go: "CVE-2025-22871"
- Dipendenze aggiuntive: "CVE-2025-23109"

Novità della versione 26.02

Scopri le novità in NetApp Trident e Trident Protect, inclusi miglioramenti, correzioni e deprecazioni.

Trident

Miglioramenti

- **Kubernetes**:
  - Aggiunto il supporto per la disponibilità generale (GA) per la concorrenza per ONTAP-NAS (solo NFS), ONTAP-SAN (iSCSI, FCP, NVMe) e Google Cloud NetApp Volumes (GCNV), promuovendo questa funzionalità dalla Tech Preview. Vedere il "Documentazione sulla scalabilità del controller" per ulteriori informazioni.
  - Aggiunto il supporto per l'autogrow del volume in Trident in base a criteri AutoGrow personalizzati e definiti dall'utente in Trident. Vedere il "Documentazione sull'espansione automatica del volume" per ulteriori informazioni.
  - Concorrenza dei nodi Trident migliorata per migliorare la scalabilità delle operazioni dei nodi per i volumi NVMe. Vedere il "Documentazione sui volumi NVMe" per ulteriori informazioni.
  - Aggiunto supporto per l'auto-tiering dei volumi NAS GCNV tramite annotazioni PVC (`tieringPolicy`, `tieringMinimumCoolingDays`, inclusa la selezione del pool e l'eredità dei cloni. Vedere il "Configura il tiering automatico per la documentazione di Google Cloud NetApp Volumes" per ulteriori informazioni.
  - Aggiunto supporto per volumi GCNV SAN block (iSCSI) utilizzando il `google-cloud-netapp-volumes-san` driver, inclusi provisioning, mappatura dei gruppi host per nodo per l'accesso LUN e operazioni di clonazione da volume. Vedere il "Documentazione sulla configurazione dei blocchi di Google Cloud NetApp Volumes" per ulteriori informazioni.
  - Aggiunto il supporto per la configurazione automatica del backend per Amazon FSx for NetApp ONTAP. Quando si crea un StorageClass che include i parametri richiesti, Trident crea automaticamente il backend corrispondente e VolumeSnapshotClass, se necessario. Vedere il "Amazon FSx for NetApp ONTAP documentazione sulla configurazione del backend" per ulteriori informazioni.
  - Aggiunto il supporto per diversi cloud Microsoft Azure, come Azure Government e Azure China, e la configurazione cloud personalizzata per i backend di Azure NetApp Files. Consultare la "Documentazione sulla configurazione del backend di Azure NetApp Files" per ulteriori informazioni.
  - Aggiunto il supporto per Kubernetes 1.35. Consultare la "Documentazione dei requisiti" per ulteriori informazioni.

Miglioramenti sperimentali

> Non utilizzare in ambienti di produzione.

- **[Anteprima tecnica]**: Aggiunto supporto per la concorrenza per i driver ONTAP-NAS-Economy e ONTAP-SAN-Economy.

Problemi noti

> C'è un problema noto con il `csi-snapshotter` sidecar. In tutte le versioni di Kubernetes, VolumeGroupSnapshots `v1beta1` impedisce a VolumeSnapshots di raggiungere lo `ReadyToUse` stato.

Sono disponibili due soluzioni alternative:

1. Eliminare i CRD VolumeGroupSnapshots per disabilitare VolumeGroupSnapshots, quindi reinstallare Trident.

2. Installa VolumeGroupSnapshots `v1beta2` e `snapshot-controller` la versione 8.4.0 o successiva, quindi reinstalla Trident. VolumeGroupSnapshots non funziona su versioni di Kubernetes precedenti alla v1.34.

Correzioni

- **Kubernetes**:
  - Risolto un problema per cui l'annullamento della pubblicazione di un clone di sola lettura rimuoveva le regole della policy di esportazione dal volume di origine nei driver ONTAP-NAS, ONTAP-NAS-Economy e Google Cloud NetApp Volumes "Issue #1086".
  - Sono state sostituite le immagini kubectl con varianti leggere basate su Alpine per evitare errori di pull in seguito alla deprecazione delle immagini pubbliche di Bitnami "Issue #1080".
  - Corretto il problema di conservazione dell'annotazione della distribuzione esistente durante l'aggiornamento di Trident "Issue #1004".
  - Consenti la clonazione tra diverse classi di archiviazione se entrambe le classi di archiviazione puntano allo stesso backend "Issue #1104".
  - Risolti i problemi di preparazione dei nodi causati da timeout in ambienti cloud con latenze di rete. Aumentati i valori di timeout per le installazioni basate sul cloud.
  - È stato risolto un problema nella creazione di LUN che causava la mancata impostazione dell'attributo del tipo di filesystem quando il processo entrava in uno stato di retry.
  - Corretto il lookup del volume dell'API REST per ignorare lo stato del volume, prevenendo falsi negativi durante le query del volume.
  - Migliorata l'efficienza del controller Trident per il driver ontap-nas-economy quando utilizzato su larga scala.
  - Imposta internalID durante l'importazione LUN nel driver ontap-san-economy.
  - Aumentati i limiti delle query di Azure Resource Graph per gestire più subnet.
  - Migliorati i timeout di clone split di CSI e ONTAP per evitare condizioni di competizione con alcune applicazioni di backup "Issue #1098", "Issue #1100".
  - Corretta la soppressione dei messaggi di errore LUKS "Issue #1069".
  - Corretta la gestione dei mapper LUKS obsoleti per i protocolli iSCSI e NVMe. La logica di pulizia avanzata impedisce errori di montaggio causati da mapper di dispositivi orfani.
- Corrette le limitazioni di scalabilità per i volumi RWX NVMe.
- Aggiornato il pacchetto OpenTelemetry-Go per correggere "CVE-2026-24051".

Trident Protect

Miglioramenti

- Trident Protect ora disabilita automaticamente le pianificazioni di protezione e annulla le operazioni in corso prima di un ripristino in-place, e le riattiva dopo il completamento del ripristino. Per ulteriori informazioni, consultare "Ripristina le applicazioni utilizzando Trident Protect".
- Aggiunto il campo `runImmediately` al CR della pianificazione e il flag CLI `--run-immediately` per attivare un backup o uno snapshot immediato al momento della creazione della pianificazione. Per ulteriori informazioni, consultare "Creare un piano di data Protection".
- Aggiunto il supporto per specificare un nome personalizzato per l'applicazione ripristinata utilizzando il `destinationApplicationName` field nella restore CR o il `--destination-app-name` CLI flag. Per

ulteriori informazioni, fare riferimento a "Ripristina le applicazioni utilizzando Trident Protect".

Correzioni

- Risolti i problemi di ripristino causati dalla creazione di pod prima che i relativi account di servizio richiesti fossero disponibili.
- Corretto `Roles` e `RoleBindings` che venivano saltati durante i ripristini delle applicazioni.
- Risolto il problema per cui il nome del cluster di origine non veniva visualizzato nell'output `tridentctl-protect get appvaultcontent` nonostante fosse configurato correttamente.
- Corretti gli errori di ripristino di Kopia che venivano ignorati a causa della mancata gestione `pipefail`.
- Risolti errori di snapshot e backup causati dai filtri delle risorse che escludevano i volumi persistenti.
- Risolto il problema del ripristino PVC errato nelle applicazioni multi-namespace con PVC con nomi identici tra namespace, che poteva causare la perdita di dati.

Novità della versione 25.10

Scopri le novità di Trident e Trident Protect, inclusi miglioramenti, correzioni e deprecazioni.

Trident

Miglioramenti

- **Kubernetes**:
  - Aggiunto supporto per snapshot di gruppi di volumi CSI con API Kubernetes per snapshot di gruppi di volumi v1beta1 per driver ONTAP-NAS NFS e ONTAP-SAN-Economy, oltre a ONTAP-SAN (iSCSI e FC). Vedere"Lavorare con gli snapshot del gruppo di volumi" .
  - Aggiunto supporto per il failover automatico del carico di lavoro con distacco forzato del volume per ONTAP-NAS e ONTAP-NAS-Economy (escluso SMB in entrambi i driver NAS) e per i driver ONTAP-SAN e ONTAP-SAN-Economy. Vedere"Automazione del failover delle applicazioni stateful con Trident" .
  - Concorrenza dei nodi Trident migliorata per una maggiore scalabilità nelle operazioni dei nodi per i volumi FCP.
  - Aggiunto il supporto ONTAP AFX per il driver ONTAP NAS. Vedere"Opzioni ed esempi di configurazione del NAS ONTAP" .
  - Aggiunto supporto per la configurazione delle richieste e dei limiti delle risorse di CPU e memoria per i contenitori Trident tramite i valori del grafico Helm e CR di TridentOrchestrator. ("Numero 1000" ,"Numero 927" ,"Numero 853" ,"Numero 592" ,"Numero 110").
  - Aggiunto supporto FC per la personalità ASAr2. Vedere"Opzioni ed esempi di configurazione DELLA SAN ONTAP" .
  - Aggiunta un'opzione per fornire le metriche Prometheus con HTTPS, anziché HTTP. Vedere"Monitor Trident" .
  - Aggiunta un'opzione `--no-rename` quando si importa un volume per mantenere il nome originale ma lasciare che Trident gestisca il ciclo di vita del volume. Vedere"Importa volumi" .
  - La distribuzione Trident ora viene eseguita con la classe di priorità critica per il cluster di sistema.
- Aggiunta un'opzione per il controller Trident per utilizzare la rete host tramite helm, operator e tridentctl ("Numero 858").

- Aggiunto il supporto QoS manuale al driver ANF, rendendolo pronto per la produzione in Trident 25.10; questo miglioramento sperimentale è stato introdotto in Trident 25.06.

Miglioramenti sperimentali

> Non utilizzare in ambienti di produzione.

- **[Anteprima tecnica]**: Aggiunto supporto per la concorrenza per ONTAP-NAS (solo NFS) e ONTAP-SAN (NVMe per ONTAP 9 unificato), oltre all'anteprima tecnica esistente per il driver ONTAP-SAN (protocolli iSCSI e FCP in ONTAP 9 unificato).

Correzioni

- **Kubernetes**:
  - È stata corretta l'incoerenza del nome del contenitore node-driver-registrar CSI standardizzando Linux DaemonSet in node-driver-registrar per far corrispondere il nome del contenitore e dell'immagine del contenitore e del DaemonSet di Windows.
  - Risolto un problema per cui le policy di esportazione per i qtree legacy non venivano aggiornate correttamente.
- **Openshift**:
  - Risolto il problema del pod del nodo Trident che non si avviava sui nodi Windows in Openshift perché SCC aveva allowHostDirVolumePlugin impostato su false ("Numero 950").
- Corretto il problema con l'API Kubernetes QPS che non veniva impostato tramite Helm ("Numero 975").
- Risolta l'impossibilità di montare un Persistent Volume Claim (PVC) basato su uno snapshot di un PVC del file system XFS basato su NVMe sullo stesso nodo Kubernetes.
- Risolto il problema di modifica dell'UUID dopo il riavvio dell'host/Docker in modalità NDVP aggiungendo nomi di sottosistema univoci/condivisi per ogni backend (ad esempio, netappdvp_subsystem).
- Corretti errori di montaggio per volumi iSCSI durante l'aggiornamento Trident da versioni precedenti alla 23.10 alla 24.10 e successive, risolvendo il problema "SANTType non valido".
- Risolto il problema per cui lo stato del backend Trident non passava da online a offline senza riavviare il controller Trident .
- Risolto il problema della condizione di gara intermittente che causava un lento ridimensionamento del PVC.
- Risolto il problema degli snapshot che non venivano ripuliti in caso di errori di clonazione del volume.
- Risolto il problema di mancata eliminazione del volume quando il percorso del dispositivo veniva modificato dal kernel.
- Risolto il problema di rimozione dello stage del volume dovuto al dispositivo LUKS già chiuso.
- Risolto il problema per cui le operazioni di archiviazione lente causavano errori ContextDeadline.
- L'operatore Trident attenderà il timeout configurabile k8s per verificare la versione Trident .

Trident Protect

NetApp Trident Protect offre funzionalità avanzate di gestione dei dati delle applicazioni che migliorano la funzionalità e la disponibilità delle applicazioni Kubernetes con stato supportate dai sistemi di storage NetApp ONTAP e dal provisioner di storage NetApp Trident CSI.

Miglioramenti

- Aggiunte annotazioni per controllare i timeout degli snapshot CR per i CR di pianificazione e backup:
  - `protect.trident.netapp.io/snapshot-completion-timeout`
  - `protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout`
  - `protect.trident.netapp.io/volume-snapshots-created-timeout`

  Vedere"Annotazioni di backup e pianificazione supportate" .
- Aggiunta un'annotazione alla pianificazione CR per configurare il timeout di associazione PVC, che verrà utilizzato dalla CR di backup: `protect.trident.netapp.io/pvc-bind-timeout-sec` . Vedere"Annotazioni di backup e pianificazione supportate" .
- Migliorato `tridentctl-protect` Elenchi di backup e snapshot con un nuovo campo per indicare gli errori di esecuzione.

Modifiche nel 25.06.2

Trident

Correzioni

- **Kubernetes**: risolto un problema critico per cui venivano rilevati dispositivi iSCSI errati durante il distacco dei volumi dai nodi Kubernetes.

Modifiche nel 25.06.1

Trident

> Per i clienti che utilizzano SolidFire, non aggiornare a 25.06.1 a causa di un problema noto durante l'annullamento della pubblicazione dei volumi. 25.06.2 verrà rilasciata a breve per risolvere questo problema.

Correzioni

- **Kubernetes**:
  - Risolto un problema per cui gli NQN non venivano controllati prima di essere rimossi dai sottosistemi.
  - Risolto un problema per cui più tentativi di chiudere un dispositivo LUKS causavano errori nello scollegamento dei volumi.
  - Corretto il problema di destage del volume iSCSI quando il percorso del dispositivo è cambiato dalla sua creazione.
  - Clonazione a blocchi di volumi tra classi di archiviazione.
- **OpenShift**: risolto un problema per cui la preparazione del nodo iSCSI non riusciva con OCP 4.19.
- Aumentato il timeout durante la clonazione di un volume utilizzando i backend SolidFire ("Problema n. 1008").

Modifiche nel 25.06

Trident

Miglioramenti

- **Kubernetes**:
  - Aggiunto supporto per snapshot di gruppi di volumi CSI con `v1beta1` API Kubernetes per snapshot di gruppi di volumi per il driver iSCSI ONTAP-SAN. Vedere "Lavorare con gli snapshot del gruppo di volumi".

    > VolumeGroupSnapshot è una funzionalità beta di Kubernetes con API beta. La versione minima richiesta per VolumeGroupSnapshot è Kubernetes 1.32.

  - Aggiunto il supporto per ONTAP ASA r2 per NVMe/TCP oltre a iSCSI. Vedere"Opzioni ed esempi di configurazione DELLA SAN ONTAP" .
  - Aggiunto il supporto SMB sicuro per i volumi ONTAP-NAS e ONTAP-NAS-Economy. Gli utenti e i gruppi di Active Directory possono ora essere utilizzati con volumi SMB per una maggiore sicurezza. Vedere "Abilita SMB sicuro".
  - Concorrenza dei nodi Trident migliorata per una maggiore scalabilità nelle operazioni dei nodi per volumi iSCSI.
  - Aggiunto `--allow-discards` quando si aprono volumi LUKS per consentire i comandi discard/TRIM per il recupero di spazio.
  - Prestazioni migliorate durante la formattazione di volumi crittografati con LUKS.
  - Pulizia LUKS avanzata per dispositivi LUKS non riusciti ma parzialmente formattati.
  - Idempotenza del nodo Trident migliorata per il collegamento e il distacco del volume NVMe.
  - Aggiunto `internalID` campo nella configurazione del volume Trident per il driver ONTAP-SAN-Economy.
  - Aggiunto supporto per la replicazione del volume con SnapMirror per backend NVMe. Vedere "Replica dei volumi con SnapMirror".

Miglioramenti sperimentali

> Non utilizzare in ambienti di produzione.

- [Anteprima tecnica] Abilitate le operazioni simultanee del controller Trident tramite `--enable-concurrency` flag di funzionalità. Ciò consente l'esecuzione parallela delle operazioni del controller, migliorando le prestazioni in ambienti affollati o di grandi dimensioni.

  > Questa funzionalità è sperimentale e attualmente supporta flussi di lavoro paralleli limitati con il driver ONTAP-SAN (protocolli iSCSI e FCP).

- [Anteprima tecnica] Aggiunto il supporto QOS manuale con il driver ANF.

Correzioni

- **Kubernetes**:
  - È stato risolto un problema con CSI NodeExpandVolume per cui i dispositivi multipath potevano avere dimensioni incongruenti quando i dischi SCSI sottostanti non erano disponibili.
  - Risolto il problema relativo all'errore di pulizia dei criteri di esportazione duplicati per i driver ONTAP-

NAS e ONTAP-NAS-Economy.

  - Risolto il problema per cui i volumi GCNV utilizzavano NFSv3 come predefinito quando `nfsMountOptions` non era impostato; ora sono supportati entrambi i protocolli NFSv3 e NFSv4. Se `nfsMountOptions` non viene fornito, viene utilizzata la versione NFS predefinita dell'host (NFSv3 o NFSv4).
  - Risolto il problema di distribuzione durante l'installazione di Trident tramite Kustomize ("Problema n. 831").
  - Corretti i criteri di esportazione mancanti per i PVC creati da snapshot ("Problema n. 1016").
  - Risolto il problema per cui le dimensioni del volume ANF non venivano automaticamente allineate con incrementi di 1 GiB.
  - Risolto il problema relativo all'utilizzo di NFSv3 con Bottlerocket.
- Risolto il problema relativo all'espansione dei volumi ONTAP-NAS-Economy fino a 300 TB nonostante gli errori di ridimensionamento.
- Risolto il problema per cui le operazioni di suddivisione del clone venivano eseguite in modo sincrono quando si utilizzava l'API REST ONTAP.

Deprecazioni:

- **Kubernetes**: aggiornato il supporto minimo di Kubernetes alla versione 1.27.

Trident Protect

NetApp Trident Protect offre funzionalità avanzate di gestione dei dati delle applicazioni che migliorano la funzionalità e la disponibilità delle applicazioni Kubernetes con stato supportate dai sistemi di storage NetApp ONTAP e dal provisioner di storage NetApp Trident CSI.

Miglioramenti

- Tempi di ripristino migliorati, offrendo la possibilità di eseguire backup completi più frequenti.
- Miglioramento della granularità della definizione dell'applicazione e ripristino selettivo con filtro Group-Version-Kind (GVK).
- Risincronizzazione efficiente e replica inversa quando si utilizza AppMirrorRelationship (AMR) con NetApp SnapMirror, per evitare la replica PVC completa.
- Aggiunta la possibilità di utilizzare EKS Pod Identity per creare bucket AppVault, eliminando la necessità di specificare un segreto con le credenziali del bucket per i cluster EKS.
- Aggiunta la possibilità di saltare il ripristino di etichette e annotazioni nello spazio dei nomi di ripristino, se necessario.
- AppMirrorRelationship (AMR) ora verificherà l'espansione del PVC di origine ed eseguirà l'espansione appropriata sul PVC di destinazione, se necessario.

Correzioni

- Risolto un bug per cui i valori di annotazione degli snapshot precedenti venivano applicati a quelli più recenti. Ora tutte le annotazioni degli snapshot vengono applicate correttamente.
- Definito un segreto per la crittografia del data mover (Kopia/Restic) per impostazione predefinita, se non definito.
- Aggiunti messaggi di convalida e di errore migliorati per la creazione di S3 AppVault.

- AppMirrorRelationship (AMR) ora replica solo i PV nello stato Bound, per evitare tentativi falliti.
- Risolto il problema per cui venivano visualizzati errori durante l'ottenimento di AppVaultContent su un AppVault con un numero elevato di backup.
- Per evitare errori, gli snapshot VMSnap di KubeVirt vengono esclusi dalle operazioni di ripristino e failover.
- Risolto il problema con Kopia per cui gli snapshot venivano rimossi prematuramente perché la pianificazione di conservazione predefinita di Kopia sovrascriveva quanto impostato dall'utente nella pianificazione.

Modifiche nel 25.02.1

Trident

Correzioni

- **Kubernetes**:
  - È stato risolto un problema nell'operatore Trident in cui i nomi e le versioni delle immagini sidecar erano compilati in modo errato quando si utilizzava un registro delle immagini non predefinito ("Problema n. 983").
  - Risolto il problema a causa del quale le sessioni multipath non riescono a recuperare durante un giveback di failover ONTAP ("Problema n. 961").

Modifiche nel 25,02

A partire da Trident 25.02, il riepilogo delle novità fornisce dettagli su miglioramenti, correzioni e deprecazioni per le versioni Trident e Trident Protect.

Trident

Miglioramenti

- **Kubernetes**:
  - Aggiunto supporto per ONTAP ASA R2 per iSCSI.
  - Aggiunto supporto per il distacco forzato di volumi ONTAP-NAS durante scenari di arresto dei nodi non regolari. I nuovi volumi ONTAP-NAS utilizzeranno ora le policy di esportazione per volume gestite da Trident. Fornito un percorso di upgrade dei volumi esistenti per passare al nuovo modello di policy di esportazione quando non vengono pubblicati, senza influire sui workload attivi.
  - Aggiunta dell'annotazione CloneFromSnapshot.
  - Aggiunto supporto per il cloning di volumi con namespace incrociato.
  - Correzioni avanzate di scansione con riparazione automatica iSCSI per avviare la nuova scansione in base all'host, al canale, alla destinazione e all'ID LUN esatti.
  - Aggiunto supporto per Kubernetes 1,32.
- **OpenShift**:
  - Aggiunto supporto per la preparazione automatica del nodo iSCSI per RHCOS sui cluster ROSA.
  - Aggiunto supporto per la virtualizzazione OpenShift per i driver ONTAP.
- Aggiunto supporto Fibre Channel su driver ONTAP-SAN.
- Aggiunto supporto NVMe LUKS.

- È stata modificata l'immagine da zero per tutte le immagini di base.
- È stato aggiunto il rilevamento e la registrazione dello stato della connessione iSCSI quando le sessioni iSCSI devono essere collegate ma non sono ("Problema n. 961").
- Aggiunto supporto per volumi SMB con il driver google-cloud-NetApp-Volumes.
- Aggiunto il supporto per consentire ai volumi ONTAP di saltare la coda di ripristino all'eliminazione.
- Aggiunto il supporto per sovrascrivere le immagini predefinite utilizzando SHA invece di tag.
- Aggiunto flag image-pull-secrets al programma di installazione tridentctl.

Correzioni

- **Kubernetes**:
  - Corretti gli indirizzi IP dei nodi mancanti dai criteri di esportazione automatica ("Problema n. 965").
  - È stato risolto il problema del passaggio prematuro delle policy di esportazione automatiche a policy per volume per ONTAP-NAS-Economy.
  - Corrette credenziali di configurazione backend per supportare tutte le partizioni AWS ARN disponibili ("Problema n. 913").
  - Aggiunta opzione per disattivare la riconciliazione del configuratore automatico nell'operatore Trident ("Problema n. 924").
  - È stato aggiunto SecurityContext per il contenitore csi-resizer ("Problema n. 976").

Trident Protect

NetApp Trident Protect offre funzionalità avanzate di gestione dei dati delle applicazioni che migliorano la funzionalità e la disponibilità delle applicazioni Kubernetes con stato supportate dai sistemi di storage NetApp ONTAP e dal provisioner di storage NetApp Trident CSI.

Miglioramenti

- Aggiunto supporto per backup e ripristino per VM di virtualizzazione KubeVirt/OpenShift per l'archiviazione volumeMode: File e volumeMode: Block (dispositivo raw). Questo supporto è compatibile con tutti i driver Trident e migliora le funzionalità di protezione esistenti durante la replica dello storage tramite NetApp SnapMirror con Trident Protect.
- Aggiunta la capacità di controllare il comportamento di congelamento a livello di applicazione per gli ambienti Kubevirt.
- Aggiunto supporto per la configurazione delle connessioni proxy AutoSupport.
- Aggiunta la possibilità di definire un segreto per la crittografia del data mover (Kopia / Restic).
- Aggiunta la possibilità di eseguire manualmente un gancio di esecuzione.
- Aggiunta la possibilità di configurare vincoli di contesto di sicurezza (SCC) durante l'installazione Trident Protect.
- Aggiunto supporto per la configurazione di nodeSelector durante l'installazione Trident Protect.
- Aggiunto il supporto per il proxy di uscita HTTP / HTTPS per gli oggetti AppVault.
- ResourceFilter esteso per consentire l'esclusione delle risorse con ambito cluster.
- Aggiunto supporto per il token di sessione AWS nelle credenziali AppVault S3.
- Aggiunto supporto per la raccolta di risorse dopo hook di esecuzione pre-snapshot.

Correzioni

- Gestione dei volumi temporanei migliorata per ignorare la coda di ripristino del volume ONTAP.
- Le annotazioni SCC vengono ora ripristinate ai valori originali.
- Maggiore efficienza di ripristino con supporto per operazioni parallele.
- Supporto avanzato per i timeout di esecuzione delle chiamate per applicazioni di grandi dimensioni.

Modifiche nel 24.10.1

Miglioramenti

- **Kubernetes**: Aggiunto il supporto per Kubernetes 1,32.
- È stato aggiunto il rilevamento e la registrazione dello stato della connessione iSCSI quando le sessioni iSCSI devono essere collegate ma non sono ("Problema n. 961").

Correzioni

- Corretti gli indirizzi IP dei nodi mancanti dai criteri di esportazione automatica ("Problema n. 965").
- È stato risolto il problema del passaggio prematuro delle policy di esportazione automatiche a policy per volume per ONTAP-NAS-Economy.
- Dipendenze Trident e Trident-ASUP aggiornate per CVE-2024-45337 e CVE-2024-45310.
- Sono state rimosse le disconnessioni per i portali non CHAP non integri in modo intermittente durante l'autoriparazione iSCSI ("Problema n. 961").

Modifiche nel 24,10

Miglioramenti

- Google Cloud NetApp Volumes driver è ora generalmente disponibile per NFS Volumes e supporta il provisioning consapevole delle zone.
- L'identità del workload GCP verrà utilizzata come identità cloud per Google Cloud NetApp Volumes con GKE.
- Aggiunto `formatOptions` parametro di configurazione ai driver ONTAP-SAN e ONTAP-SAN-Economy per consentire agli utenti di specificare le opzioni di formato LUN.
- Dimensioni minime del volume Azure NetApp Files ridotte a 50 GiB. È prevista la disponibilità di Azure di nuove dimensioni minime per novembre.
- Aggiunto `denyNewVolumePools` parametro di configurazione per limitare i driver ONTAP-NAS-Economy e ONTAP-SAN-Economy ai pool FlexVol preesistenti.
- Aggiunto rilevamento per aggiunta, rimozione o ridenominazione di aggregati dalla SVM in tutti i driver ONTAP.
- Aggiunti 18 MiB di overhead ai LUN LUKS per garantire che le dimensioni PVC segnalate siano utilizzabili.
- Miglioramento dello stadio del nodo ONTAP-SAN e ONTAP-SAN-Economy e annullamento della gestione degli errori per consentire l'annullamento della rimozione dei dispositivi dopo una fase di guasto.
- È stato aggiunto un generatore di ruoli personalizzato che consente ai clienti di creare un ruolo minimalista per Trident in ONTAP.
- Aggiunta ulteriore registrazione per la risoluzione dei problemi `lsscsi` ("Problema n. 792").

Kubernetes

- Aggiunta di nuove funzionalità Trident per i flussi di lavoro nativi per Kubernetes:
  - Protezione dei dati
  - Migrazione dei dati
  - Disaster recovery
  - Mobilità delle applicazioni

    "Scopri di più su Trident Protect".
- Aggiunta una nuova bandiera `--k8s-api-qps` agli installatori per impostare il valore QPS utilizzato da Trident per comunicare con il server API Kubernetes.
- Aggiunto `--node-prep` flag agli installatori per la gestione automatica delle dipendenze del protocollo storage per i nodi del cluster Kubernetes. Compatibilità testata e verificata con il protocollo storage iSCSI Amazon Linux 2023
- Aggiunto supporto per il distacco forzato per volumi ONTAP-NAS-Economy durante scenari di spegnimento nodi non-Graceful.
- I nuovi volumi NFS ONTAP-NAS-Economy utilizzeranno le policy di esportazione per qtree quando si utilizza `autoExportPolicy` l'opzione backend. I qtree verranno mappati solo alle policy di esportazione restrittive dei nodi al momento della pubblicazione, per migliorare il controllo degli accessi e la sicurezza. Le qtree esistenti passeranno al nuovo modello di policy di esportazione quando Trident pubblica il volume da tutti i nodi per farlo senza impatti sui carichi di lavoro attivi.
- Aggiunto supporto per Kubernetes 1,31.

Miglioramenti sperimentali

- Aggiunta dell'anteprima tecnica per il supporto Fibre Channel su driver ONTAP-SAN.

Correzioni

- **Kubernetes**:
  - Gancio a nastro per l'ammissione del Rancher fisso che impedisce l'installazione di Trident Helm ("Problema n. 839").
  - Chiave di affinità fissa nei valori del grafico del timone ("Problema n. 898").
  - Fixed tridentControllerPluginNodeSelector/tridentNodePluginNodeSelector non funziona con il valore "true" ("Problema n. 899").
  - Sono stati eliminati gli snapshot effimeri creati durante la clonazione ("Problema n. 901").
- Aggiunto supporto per Windows Server 2019.
- Corretto `go mod Tidy`in Trident repo ("Problema n. 767").

Dipendenze

- **Kubernetes:**
  - Aggiornato il numero minimo di Kubernetes supportati a 1,25.
  - Rimosso il supporto per i criteri di protezione POD.

Rebranding dei prodotti

A partire dalla release 24,10, Astra Trident viene rinominato Trident (NetApp Trident). Il rebranding non influisce su funzionalità, piattaforme supportate o interoperabilità per Trident.

Modifiche nel 24,06

Miglioramenti

- **IMPORTANTE**: Il `limitVolumeSize` parametro ora limita le dimensioni di qtree/LUN nei driver ONTAP economy. Utilizzare il nuovo `limitVolumePoolSize` parametro per controllare le dimensioni FlexVol in tali driver. ("Problema n. 341").
- È stata aggiunta la capacità di autoriparazione iSCSI di avviare scansioni SCSI con l'ID LUN esatto se sono in uso igroup deprecati ("Problema n. 883").
- Supporto aggiunto per le operazioni di cloning e ridimensionamento del volume da consentire anche quando il backend è in modalità sospesa.
- È stata aggiunta la possibilità di propagare ai pod di nodi Trident le impostazioni di registro configurate dall'utente per il controller Trident.
- È stato aggiunto il supporto in Trident per l'utilizzo di REST per impostazione predefinita invece di ONTAPI (ZAPI) per ONTAP versioni 9.15.1 e successive.
- Aggiunto supporto per nomi di volumi e metadati personalizzati sui backend di storage ONTAP per nuovi volumi persistenti.
- Migliorato il `azure-netapp-files` driver (ANF) per abilitare automaticamente la directory snapshot per impostazione predefinita quando le opzioni di montaggio NFS sono impostate per utilizzare NFS versione 4.x
- Aggiunto supporto Bottlerocket per volumi NFS.
- Aggiunto il supporto dell'anteprima tecnica per Google Cloud NetApp Volumes.

Kubernetes

- Aggiunto supporto per Kubernetes 1,30.
- Aggiunta la possibilità per Trident DaemonSet di pulire i montaggi zombie e i file di tracciamento residui all'avvio ("Problema n. 883").
- Aggiunta annotazione PVC `trident.netapp.io/luksEncryption` per l'importazione dinamica dei volumi LUKS ("Problema n. 849").
- Aggiunta della conoscenza della topologia al driver ANF.
- Aggiunto supporto per nodi Windows Server 2022.

Correzioni

- Risolti i problemi di installazione di Trident a causa di transazioni obsolete.
- Corretto tridentctl per ignorare i messaggi di avviso da Kubernetes ("Problema n. 892").
- La priorità del controller Trident è stata modificata `SecurityContextConstraint` in `0` ("Problema n. 887").
- I driver ONTAP ora accettano dimensioni di volume inferiori a 20 MiB ("Problema[#885").
- Trident fisso per impedire la riduzione dei volumi FlexVol durante l'operazione di ridimensionamento per il driver ONTAP-SAN.

- Risolto un errore di importazione del volume ANF con NFS v4,1.

Modifiche nel 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 supporto per installare 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 Amazon FSX ai driver ONTAP per consentire l'integrazione con AWS IAM e SecretsManager e per consentire a 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 Trident su un nodo Windows quando il nodo viene riavviato.

Miglioramenti

Kubernetes

Aggiunto supporto per Kubernetes 1,28.

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").

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").

Trident

- Richieste ONTAPI (ZAPI) fisse per garantire che i numeri di serie LUN vengano interrogati quando si ottengono attributi LUN per identificare e correggere 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").

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 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.

Trident

- Aggiunto supporto per volumi SMB con Amazon FSX per NetApp 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.
- Migliori performance del driver di storage ONTAP-nas nella gestione di più volumi 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.
- DataLIF non deve più essere specificato durante la configurazione `ontap-san` o `ontap-san-economy` i driver.

Cambiamenti nel 22.10

Prima di eseguire l'aggiornamento a Trident 22,10, è necessario leggere le seguenti informazioni critiche.

** informazioni aggiornate su Trident 22.10 **

- Kubernetes 1,25 è ora supportato in Trident. Devi eseguire l'aggiornamento di Trident alla versione 22,10 prima di eseguire l'aggiornamento a Kubernetes 1,25.
- Trident ora applica rigorosamente l'utilizzo della configurazione multipath negli ambienti SAN, con un valore consigliato di `find_multipaths: no` 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 di SLM specifico dei backend SAN ONTAP per garantire la pubblicazione solo di un sottoinsieme di LIF dati 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 i tentativi granulari nel flusso di lavoro iSCSI Trident per fallire rapidamente 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 eseguire l'aggiornamento di 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 Trident stabilirà 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 alle semplici informazioni host e al rilevamento attivo del servizio per confermare al meglio che i servizi NFS/iSCSI sono disponibili sui nodi di lavoro.

Documentazione

È stata aggiunta una nuova "Standard di sicurezza Pod"sezione (PSS) con i dettagli delle autorizzazioni abilitate da Trident all'installazione.

Cambiamenti nel 22.04

NetApp continua a migliorare e migliorare i propri prodotti e servizi. Ecco alcune delle funzioni più recenti di Trident. Per le versioni precedenti, fare riferimento alla "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.
- Ora i dati LIF SAN ONTAP IPv6 funzionano se specificati con parentesi.
- 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 quando le performance di Trident rallentano quando vengono creati > 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 Trident ha registrato un errore irreversibile durante l'arresto (problema 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 a causa del quale tutti i programmi di installazione di Trident potevano identificare Kubernetes semplice come OpenShift (GitHub problema 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 da 1,24 a 1,25 o versione successiva su `true` cui è installato Trident, è necessario aggiornare Values.yaml per impostarlo `excludePodSecurityPolicy` o aggiungerlo `--set excludePodSecurityPolicy=true` al `helm upgrade` comando prima di poter aggiornare il cluster.
- Trident ora applica uno spazio vuoto `fsType` (`fsType=""`) per i volumi che non hanno lo `fsType` specificato nella classe StorageClass. Quando si utilizza Kubernetes 1,17 o versione successiva, Trident supporta l'offerta di un bianco `fsType` per i volumi NFS. Per i volumi iSCSI, è necessario impostare `fsType` su StorageClass quando si applica un utilizzo di un `fsGroup` contesto di protezione.
- Quando si utilizza un backend tra più istanze Trident, ogni file di configurazione backend deve avere un `storagePrefix` valore diverso per i backend ONTAP o utilizzare un valore diverso `TenantName` per i backend SolidFire. Trident non è in grado di rilevare volumi creati da altre istanze di Trident. Il tentativo di creare un volume esistente sui backend ONTAP o SolidFire ha esito positivo, poiché Trident considera la creazione di volume come un'operazione idempoter. Se `storagePrefix` o `TenantName` non differiscono, potrebbero esserci collisioni di nomi per i volumi creati sullo stesso backend.
- Quando si installa Trident (utilizzando `tridentctl` o l'operatore Trident) e si utilizza `tridentctl` per gestire Trident, è necessario assicurarsi che la `KUBECONFIG` variabile di ambiente sia impostata. Ciò è necessario per indicare il cluster Kubernetes `tridentctl` con cui dovrebbe lavorare. Quando si lavora con più ambienti Kubernetes, occorre assicurarsi che il `KUBECONFIG` file sia 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 è vero per le istanze RHEL/Red Hat Enterprise Linux CoreOS (RHCOS), che richiedono `discard` "opzione di montaggio" ; assicurarsi che l'opzione Discard mountOption sia inclusa in[`StorageClass` ^] per supportare l'eliminazione dei blocchi online.
- Se disponi di più di un'istanza di Trident per cluster Kubernetes, Trident non può comunicare con altre istanze e non può rilevare altri volumi che hanno creato, il che porta a un comportamento imprevisto e non corretto se vengono eseguite più istanze all'interno di un cluster. Dovrebbe esserci una sola istanza di Trident per cluster Kubernetes.

- Se gli oggetti basati su Trident `StorageClass` vengono eliminati da Kubernetes mentre Trident è offline, Trident non rimuove le classi di storage corrispondenti dal proprio database quando torna online. È necessario eliminare queste classi di archiviazione utilizzando `tridentctl` o l'API REST.
- Se un utente elimina un PV fornito da Trident prima di eliminare il PVC corrispondente, Trident non elimina automaticamente il volume di backup. È necessario 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 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]`.

> Non è possibile specificare `dataLIF` su un backend SAN ONTAP. Trident scopre 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

- "Trident GitHub"
- "Blog Trident"

Versioni precedenti della documentazione

Se non si utilizza Trident 26.10, la documentazione per le versioni precedenti è disponibile in base al "Ciclo di vita del supporto Trident".

- "Trident 26.06"
- "Trident 26.02"
- "Trident 25.10"
- "Trident 25.06"
- "Trident 25,02"
- "Trident 24,10"
- "Trident 24,06"
- "Trident 24,02"
- "Trident 23,10"
- "Trident 23,07"
- "Trident 23,04"
- "Trident 23,01"

NetApp Trident supporto per sistemi di storage ONTAP ASA r2

NetApp Trident 25.02 e versioni successive supportano i sistemi NetApp ASA r2 come backend di storage. Fare riferimento a "Sistemi ASA r2" per ulteriori informazioni.

I sistemi ASA r2 richiedono il `ontap-san` driver. Trident non supporta il `ontap-san-economy` driver per i sistemi ASA r2.

Quando si specifica `ontap-san` come `storageDriverName` nella configurazione backend, Trident rileva automaticamente il sistema storage ASA r2.

Trident fornisce una protezione dati limitata per i sistemi ASA r2 con Trident protect.

I protocolli SAN supportati dipendono dalla versione di Trident:

- Trident 25.02 e versioni successive supporta iSCSI.
- Trident 25.06 e versioni successive supportano NVMe/TCP oltre a iSCSI.
- Trident 25.10 e versioni successive supportano FC oltre a iSCSI e NVMe/TCP.

È necessario assegnare almeno un aggregato alla macchina virtuale di storage (SVM) per lo storage back-end ONTAP. Consultare "Assegnazione di aggregati a SVM nei sistemi ASA r2" per le istruzioni.

Operazioni supportate

- Provisioning di volumi persistenti (PVs)
- Provisioning dinamico del volume
- Creazione ed eliminazione di volumi
- Clonazione dei volumi
- Espansione dei volumi
- Gestione delle classi di archiviazione

Operazioni non supportate

- Crittografia Linux Unified Key Setup (LUKS)
- Replica di volumi SnapMirror
- Limitazione dell'utilizzo aggregato
- Modalità di prenotazione dello spazio
- Snapshot
- Tiering

Per ulteriori informazioni, fare riferimento a "Opzioni ed esempi di configurazione DELLA SAN ONTAP" .

Problemi noti

I problemi noti identificano i problemi che potrebbero impedire l'utilizzo corretto di questa

versione del prodotto.

I seguenti problemi noti riguardano la versione corrente:

I volumi rimangono collegati dopo l'aggiornamento da Trident 22.10 o versioni precedenti

Questo problema riguarda i clienti che eseguono l'aggiornamento direttamente da Trident 22.10 o versioni precedenti a Trident 26.10 con i volumi collegati. Gli aggiornamenti da Trident 23.01 o versioni successive non sono interessati.

Dopo l'aggiornamento, i volumi interessati potrebbero non scollegarsi. I pod che utilizzano volumi ReadWriteOnce (RWO) non possono essere spostati su un altro nodo e potrebbero segnalare un errore di multi-attach. Anche l'eliminazione del PersistentVolume (PV) potrebbe bloccarsi.

Per ovviare a questo problema, sul nodo che presenta il collegamento obsoleto del volume, elimina il file residuo `volumePublishInfo.json` dalla directory `globalmount` di quel volume.

VolumeSnapshots non raggiungono lo stato ReadyToUse

> C'è un problema noto con il `csi-snapshotter` sidecar. In tutte le versioni di Kubernetes, VolumeGroupSnapshots `v1beta1` impedisce a VolumeSnapshots di raggiungere lo `ReadyToUse` stato.

Sono disponibili due soluzioni alternative:

1. Eliminare i CRD VolumeGroupSnapshots per disabilitare VolumeGroupSnapshots, quindi reinstallare Trident.
2. Installa VolumeGroupSnapshots `v1beta2` e `snapshot-controller` la versione 8.4.0 o successiva, quindi reinstalla Trident. VolumeGroupSnapshots non funziona su versioni di Kubernetes precedenti alla v1.34.

Il ripristino dei backup di file di grandi dimensioni può non riuscire

Quando si ripristinano file da 30 GB o più da un backup Amazon S3 effettuato tramite Restic, l'operazione di ripristino potrebbe non riuscire. Come soluzione alternativa, esegui il backup dei dati utilizzando Kopia come strumento di spostamento dati (Kopia è lo strumento di spostamento dati predefinito per i backup). Fare riferimento a "Proteggi le applicazioni utilizzando Trident Protect" per istruzioni.

Inizia subito

Scopri Trident

Scopri Trident

Trident è un progetto open source completamente supportato gestito da NetApp. 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 Trident?

NetApp Trident consente l'utilizzo e la gestione delle risorse di storage su tutte le piattaforme di storage NetApp più diffuse, nel cloud pubblico o on-premises, inclusi i cluster ONTAP on-premises (AFF, FAS e ASA), ONTAP Select, Cloud Volumes ONTAP, Element software (NetApp HCI, SolidFire), Azure NetApp Files, Amazon FSx for NetApp ONTAP e Google Cloud NetApp Volumes.

Trident è un orchestrator di storage dinamico conforme a Container Storage Interface (CSI) che si integra in modo nativo con "Kubernetes". Trident funziona come un singolo pod controller e un pod nodo su ciascun nodo di lavoro nel cluster. Per ulteriori informazioni, fare riferimento alla "Architettura Trident" sezione.

Trident fornisce anche un'integrazione diretta con l'ecosistema Docker per le piattaforme 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. Per ulteriori informazioni, fare riferimento alla "Implementa Trident per Docker" sezione.

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

Piattaforme Kubernetes supportate

Trident è supportato su una gamma di distribuzioni e piattaforme Kubernetes.

Le piattaforme supportate includono: * Kubernetes upstream * Red Hat OpenShift * SUSE Harvester 1.7.0 (ONTAP iSCSI)

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.

▼ **Google Cloud NetApp Volumes**

"Google Cloud NetApp Volumes" È un servizio di file storage completamente gestito in Google Cloud che offre file storage Enterprise dalle performance elevate.

▼ **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.

Trident è in grado di eseguire il provisioning e la gestione dei dispositivi di storage per le applicazioni containerizzate direttamente sulla piattaforma di storage NetApp HCI sottostante.

▼ **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 dispongono di configurazioni all-flash, ibride o all-HDD e offrono diversi modelli di implementazione: Cluster FAS, AFA e ASA on-premise, ONTAP Select e Cloud Volumes ONTAP. Trident supporta questi modelli di implementazione ONTAP.

Architettura Trident

Trident funziona come un singolo pod controller e un pod nodo su ciascun nodo di lavoro nel cluster. Il pod di nodo deve essere in esecuzione su qualsiasi host in cui si desidera montare potenzialmente un volume Trident.

Comprensione dei pod controller e dei pod di nodi

Trident implementa come singolo Pod controller Trident e uno o più Pod di nodi Tridentnel 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" sono utilizzati per vincolare un pod all'esecuzione su un nodo specifico o preferito. È possibile configurare selettori di nodo e tolleranze per controller e pod di nodo durante l'installazione di 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.

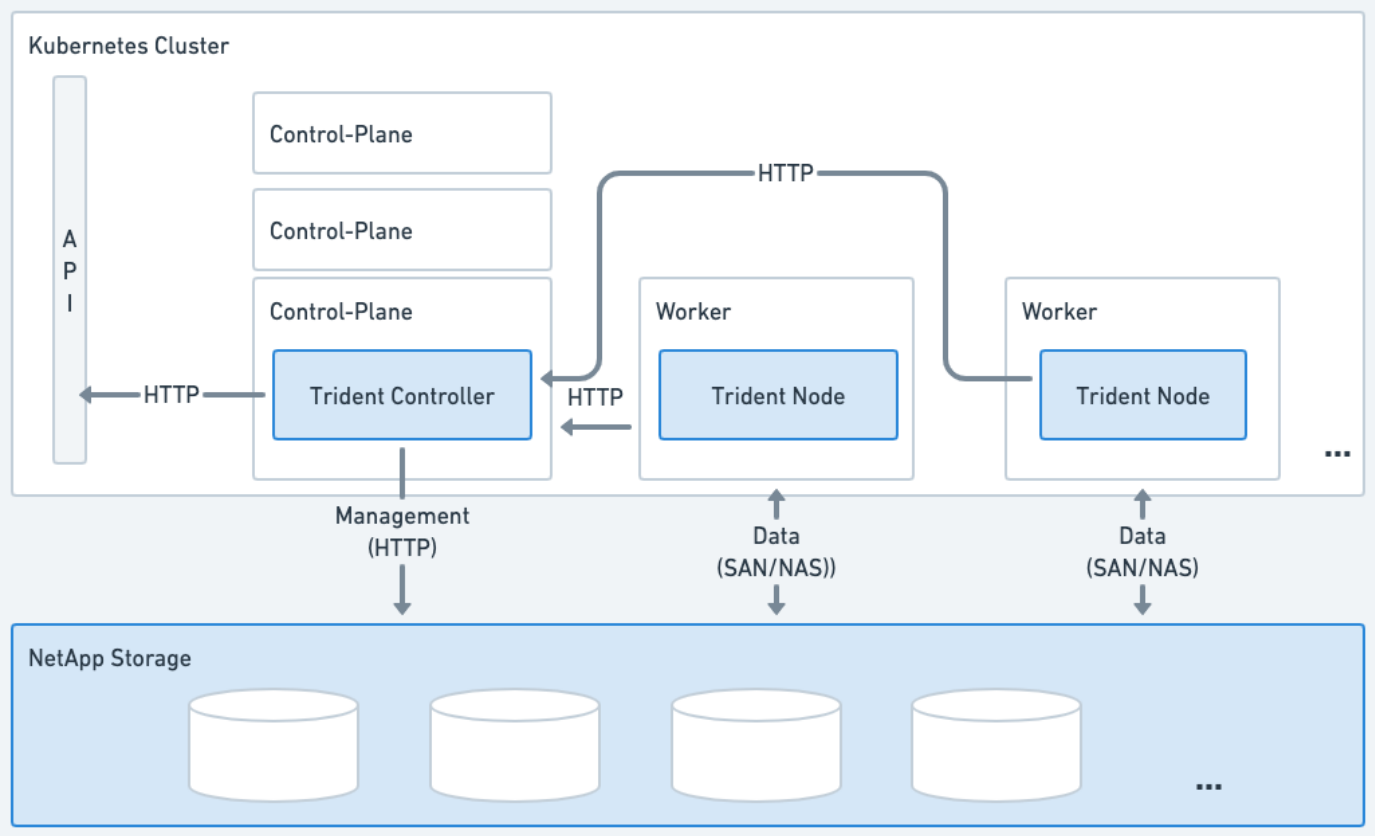


Figura 1. Trident implementato nel cluster Kubernetes

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.

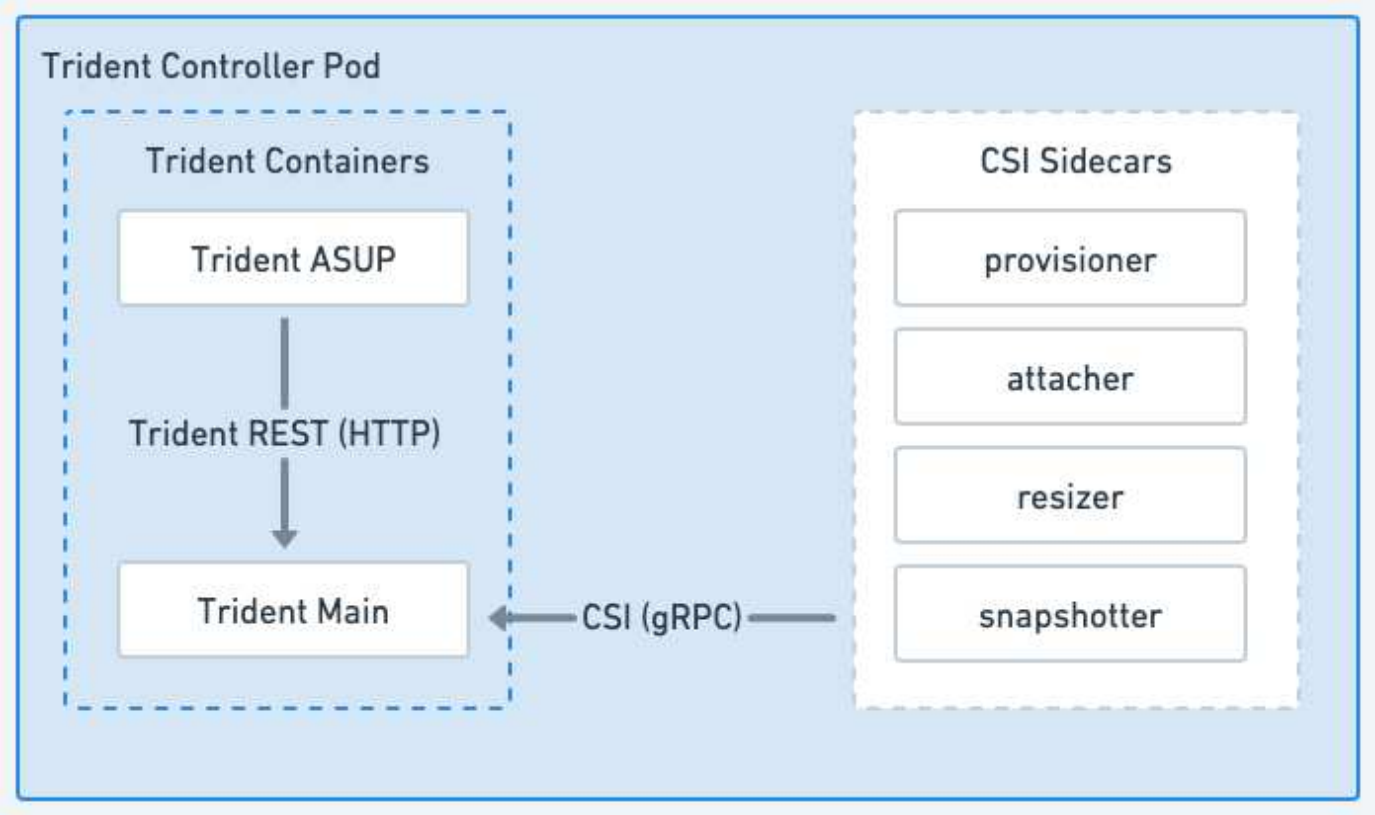


Figura 2. Diagramma del pod controller Trident

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

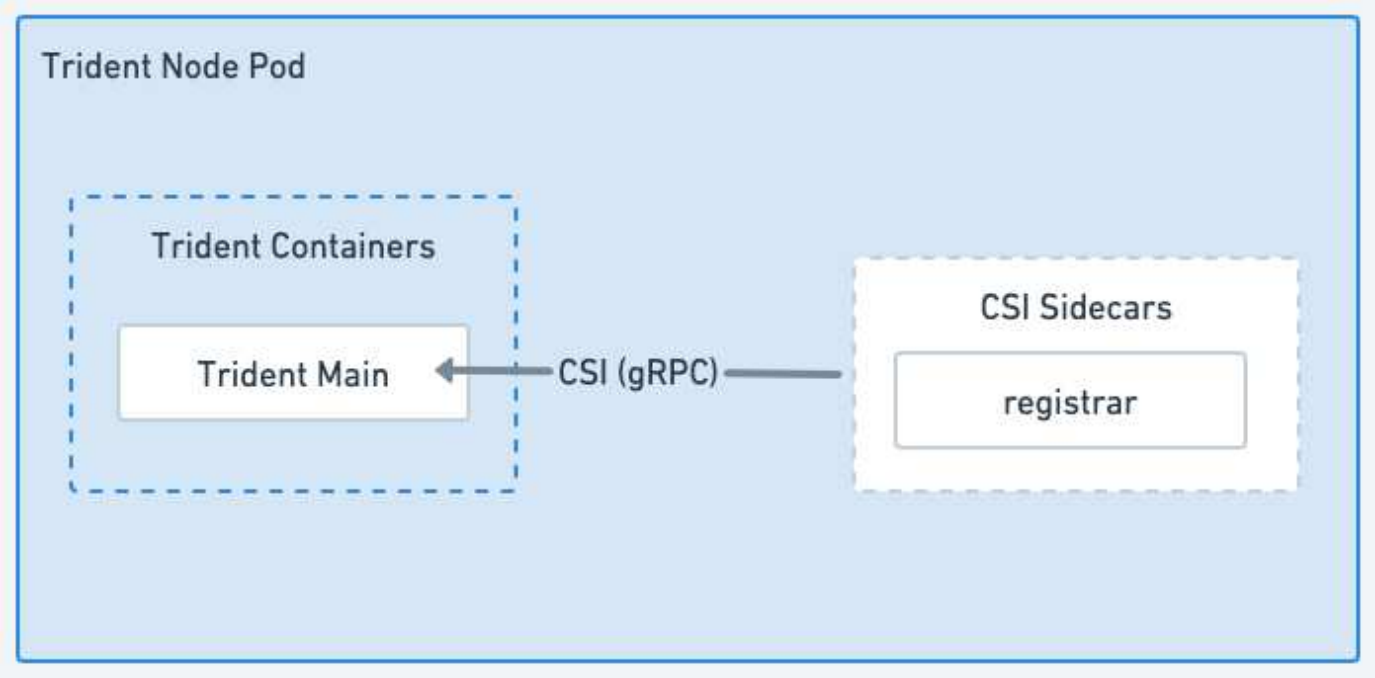


Figura 3. Diagramma del pod nodo Trident

Architetture cluster Kubernetes supportate

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

Processo di provisioning dei volumi di Trident

Il provisioning in 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 di pool di archiviazione backend a una classe di archiviazione si basa sia sugli attributi richiesti della classe di archiviazione sia sugli `storagePools` elenchi , `additionalStoragePools` e `excludeStoragePools` . 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, Trident aggiunge tale pool di storage all'insieme di pool di storage adatti per tale classe di storage. Inoltre, Trident aggiunge al set tutti i pool di storage

`additionalStoragePools` elencati, anche se i relativi attributi non soddisfano tutti o alcuni degli attributi richiesti dalla classe di storage. È necessario utilizzare l' `excludeStoragePools`elenco per ignorare e rimuovere i pool di archiviazione da utilizzare per una classe di archiviazione. Ogni volta che si aggiunge un nuovo backend, Trident esegue un processo simile, controllando se i propri pool di storage soddisfano quelli delle classi di storage esistenti e rimuovendo quelli contrassegnati come esclusi.

Creazione di volumi

Quindi, Trident utilizza le associazioni tra classi di storage e pool di storage per determinare dove eseguire il provisioning dei volumi. Quando si crea un volume, Trident ottiene per primo il set di pool di storage per la classe di storage di tale volume e, se si specifica un protocollo per il volume, 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). Trident casualmente crea l'ordine del set risultante, per facilitare una distribuzione uniforme dei volumi, quindi esegue un'iterazione, tentando di eseguire il provisioning del volume su ogni pool di storage, a sua volta. Se riesce su uno, ritorna con successo, registrando gli eventuali errori riscontrati nel processo. Trident restituisce un errore **solo se** non riesce a fornire su **tutti** i pool di archiviazione disponibili per la classe e il protocollo di archiviazione richiesti.

Creazione di snapshot di volume con Trident

Scopri di più su come Trident gestisce la creazione di snapshot di volume per i driver.

Scopri di più sulla creazione di snapshot di volumi

- Per il `ontap-nas` , `ontap-san` , E `azure-netapp-files` driver, ogni Persistent Volume (PV) viene mappato su un FlexVol volume. Di conseguenza, gli snapshot del volume vengono creati come snapshot NetApp . La tecnologia snapshot NetApp offre maggiore stabilità, scalabilità, recuperabilità e prestazioni rispetto alle tecnologie snapshot della concorrenza. Queste copie snapshot sono estremamente efficienti sia in termini di tempo necessario per crearle sia di spazio di archiviazione.
- 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 il `ontap-san-economy` driver, i PV vengono mappati alle LUN create su volumi FlexVol condivisi VolumeSnapshot di PVS vengono ottenuti eseguendo FlexClone della LUN associata. 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` i driver e `ontap-san`, le snapshot ONTAP sono copie point-in-time della FlexVol e consumano spazio sulla FlexVol stessa. 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 VolumeSnapshot creato tramite Kubernetes viene eliminato, Trident eliminerà lo snapshot ONTAP associato. È possibile eliminare anche gli snapshot ONTAP non creati tramite Kubernetes.

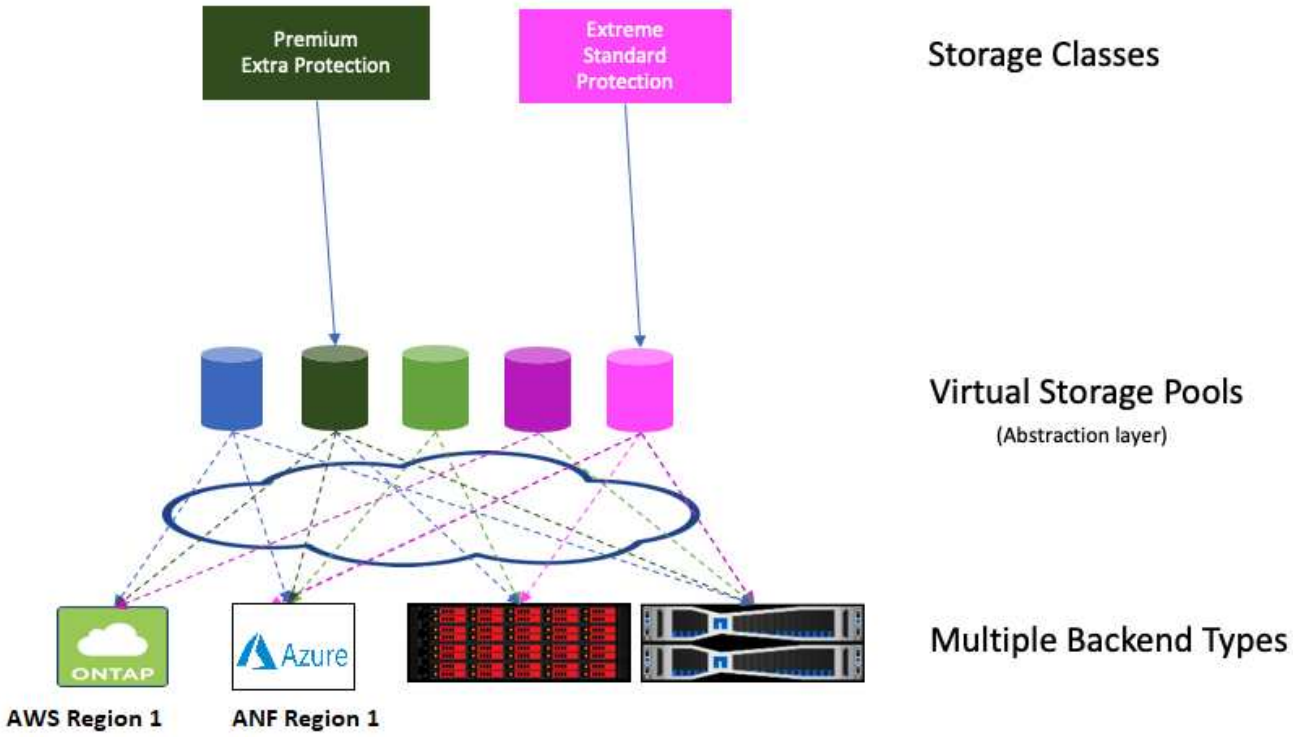
Con Trident, puoi usare VolumeSnapshots per creare nuovi PV da essi. La creazione di PV da questi snapshot viene eseguita utilizzando la tecnologia FlexClone per i backend ONTAP supportati. Quando si crea un PV da uno snapshot, il volume di supporto è un FlexClone del volume padre dello snapshot. IL `solidfire-san` il driver utilizza cloni di volume del software Element per creare PV da snapshot. Qui viene creato un clone dallo snapshot dell'Elemento.

Configurazione del pool di storage virtuale Trident

I pool virtuali forniscono un layer di astrazione tra i backend dello storage Trident e Kubernetes `StorageClasses`. Essi consentono agli amministratori di definire aspetti, quali posizione, performance e protezione per ogni backend in modo comune e indipendente dal backend, senza specificare il tipo di backend `StorageClass` 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 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.

Le etichette del pool virtuale possono essere definite utilizzando questi caratteri:

- lettere maiuscole `A-Z`
- lettere minuscole `a-z`
- numeri `0-9`
- sottolineature `_`
- trattini `-`

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

Gestione del gruppo di accesso al volume Trident

Ulteriori informazioni sull'utilizzo di 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 Trident in modalità CSI, è possibile ignorare questa sezione. Trident utilizza CHAP quando viene installato come provisioner CSI avanzato.

Informazioni sui gruppi di accesso ai volumi

Trident può utilizzare i gruppi di accesso ai volumi per controllare l'accesso ai volumi forniti. Se CHAP è disattivato, si prevede di trovare un gruppo di accesso chiamato `trident` a meno che non si specifichino uno o più ID del gruppo di accesso nella configurazione.

Trident associa nuovi volumi ai gruppi di accesso configurati, ma non crea né gestisce direttamente i gruppi di accesso. I gruppi di accesso devono esistere prima che il backend dello storage venga aggiunto a 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 Trident

È possibile installare Trident e iniziare a gestire le risorse di storage in pochi passaggi. Prima di iniziare, consultare "Requisiti Trident".

Per Docker, fare riferimento alla sezione "Trident per Docker".

1 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"

2 Installare Trident

Trident offre diversi metodi e modalità di installazione ottimizzati per una varietà di ambienti e organizzazioni.

"Installare Trident"

3 Creare un backend

Un backend definisce la relazione tra Trident e un sistema di storage. Spiega a Trident come comunicare con quel sistema storage e come Trident dovrebbe eseguire il provisioning dei volumi da esso.

"Configurare un backend" per il tuo sistema storage

4 Creare una classe di storage Kubernetes

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

"Creare una classe di storage"

5 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 Trident

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

Informazioni critiche su Trident

È necessario leggere le seguenti informazioni critiche su Trident.

**▼ informazioni aggiornate su Trident **

- Trident 26.10 supporta Kubernetes dalla versione 1.27 alla 1.37. Prima di aggiornare Kubernetes, consulta l'elenco delle immagini specifiche per la versione riportato di seguito.
- Trident impone rigorosamente l'uso della configurazione multipath 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.

Frontend supportati (orchestratori)

Trident supporta molteplici motori e Orchestrator per container, tra cui:

- Anthos on-premise (VMware) e anthos su Bare Metal 1,16
- Kubernetes 1.27 - 1.37
- OpenShift 4.12, 4.14 - 4.22 (Se prevedi di utilizzare la preparazione del nodo iSCSI con OpenShift 4.19, la versione minima di Trident supportata è 25.06.1.)

> Trident continua a supportare le versioni precedenti di OpenShift in linea con"Ciclo di vita della versione Red Hat Extended Update Support (EUS)" , anche se si basano su versioni di Kubernetes che non sono più ufficialmente supportate a monte. In questi casi, durante l'installazione Trident , puoi tranquillamente ignorare eventuali messaggi di avviso relativi alla versione di Kubernetes.

- Rancher Kubernetes Engine 2 (RKE2) v1.28.x - 1.37.x

- Talos OS per NFSv4, iSCSI e NVMe/TCP. Per ciascun protocollo sono necessarie le seguenti estensioni di sistema di Talos; verifica questi requisiti con Sidero Labs.
  - NFSv4: `siderolabs/nfs-utils`
  - iSCSI: `siderolabs/iscsi-tools`, `siderolabs/multipath-tools`, e `siderolabs/trident-iscsi-tools`
  - NVMe/TCP: `siderolabs/nvme-cli`, `siderolabs/util-linux-tools`, e `siderolabs/trident-iscsi-tools`

Trident funziona anche con un host delle altre offerte Kubernetes completamente gestite e gestite in autonomia, tra cui Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service (AKS), Mirantis Kubernetes Engine (MKE) e il portfolio VMware Tanzu.

Trident e ONTAP possono essere utilizzati come provider di archiviazione per "KubeVirt".

> Prima di aggiornare un cluster Kubernetes dalla versione 1,25 alla 1,26 o successiva in cui è installato Trident, fare riferimento alla "Aggiornare un'installazione Helm".

Back-end supportati (storage)

Per utilizzare Trident, è necessario uno o più dei seguenti backend supportati:

- Amazon FSX per NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Google Cloud NetApp Volumes
- Array All SAN (ASA) NetApp
- FAS, AFF o ASA r2 (iSCSI, NVMe/TCP e FC) in locale che eseguono versioni ONTAP con supporto NetApp completo o limitato. Vedere "Supporto della versione software".
- Software NetApp HCI/Element 11 o superiore

Supporto Trident per KubeVirt e OpenShift Virtualization

Driver di archiviazione supportati:

Trident supporta i seguenti driver ONTAP per la virtualizzazione KubeVirt e OpenShift:

- ontap-nas
- ONTAP-san (iSCSI, FCP, NVMe su TCP)
- ONTAP-san-Economy (solo iSCSI)

Punti da considerare:

- Aggiornare la classe di archiviazione in modo che il `fsType` parametro (ad esempio: `fsType: "ext4"`) Nell'ambiente di virtualizzazione OpenShift. Se necessario, impostare la modalità volume in modo da bloccare esplicitamente utilizzando il `volumeMode=Block` parametro in `dataVolumeTemplates` per notificare a CDI la creazione di volumi di dati di blocco.
- *Modalità di accesso RWX per i driver di storage a blocchi*: I driver ONTAP-san (iSCSI, NVMe/TCP, FC) e ONTAP-san-Economy (iSCSI) sono supportati solo con "volumeMode: Block" (dispositivo raw). Per questi driver, il `fstype` parametro non può essere utilizzato perché i volumi sono forniti in modalità dispositivo

raw.

- Per i flussi di lavoro di migrazione in tempo reale in cui è richiesta la modalità di accesso RWX, sono supportate le seguenti combinazioni:
  - NFS + `volumeMode=Filesystem`
  - ISCSI + `volumeMode=Block` (dispositivo raw)
  - NVMe/TCP + `volumeMode=Block` (dispositivo raw)
  - FC + `volumeMode=Block` (dispositivo raw)

Supporto Trident per Kubernetes su Kubernetes e piani di controllo ospitati OpenShift

Trident certifica pienamente la topologia Kubernetes su Kubernetes. Cluster Kubernetes i cui nodi vengono eseguiti come macchine virtuali KubeVirt su un cluster dell'infrastruttura (host). Questa topologia è alla base di offerte di control plane ospitati, come gli hosted control planes (HCP) di OpenShift su OpenShift Virtualization.

Questa configurazione funzionava nelle versioni precedenti, ma la qualifica completa è iniziata con la 26.06.

Il cluster guest (tenant) utilizza il driver CSI KubeVirt (`kubevirt-csi` come provider di storage. Quando un carico di lavoro nel cluster guest richiede un `PersistentVolumeClaim`, il driver CSI KubeVirt crea un volume corrispondente nel cluster di infrastruttura. Trident esegue il provisioning del volume da ONTAP e il volume viene collegato a caldo alla VM del nodo guest. Trident viene eseguito solo nel cluster di infrastruttura. Il cluster guest utilizza lo storage ONTAP indirettamente tramite il driver CSI KubeVirt.

Requisiti:

- Trident è installato sul cluster dell'infrastruttura con un backend ONTAP supportato.
- Esiste una storage class Trident nel cluster di infrastruttura per il driver CSI KubeVirt su cui mappare i volumi del cluster guest.
- KubeVirt o OpenShift Virtualization viene distribuito sul cluster di infrastruttura e il driver CSI KubeVirt viene installato in ogni cluster guest.

Componente	Versione
Trident	26,10
Kubernetes (infrastruttura e cluster guest)	1,37
KubeVirt	1.9.0
Importatore di dati containerizzato (CDI)	1,65

Punti da considerare:

- Nel guest cluster, i volumi appaiono come dispositivi a blocchi hot-plug sulle VM del nodo guest.
- Il flusso di lavoro del driver CSI KubeVirt non richiede Trident nel cluster guest. Configuri il backend ONTAP e la classe di storage nel cluster di infrastruttura.

Installa Trident direttamente nel cluster guest:

Puoi anche installare Trident direttamente nel cluster tenant. Installa Trident nel cluster tenant quando hai bisogno dell'intera gamma di funzionalità di Trident, come snapshot, cloni, ridimensionamento dei volumi e importazione, che `kubevirt-csi`non sono disponibili di default.

Nel cluster guest, usa protocolli basati su IP: NFS, iSCSI e NVMe/TCP. Il cluster guest non supporta Fibre Channel (FC) perché i nodi guest sono macchine virtuali senza connettività FC diretta.

Requisiti delle funzionalità

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

Funzione	Versione di Kubernetes	Sono richiesti i gate delle funzionalità?
Trident	1.27 - 1.37	No
Snapshot dei volumi	1.27 - 1.37	No
PVC dalle istantanee dei volumi	1.27 - 1.37	No
Ridimensionamento di iSCSI PV	1.27 - 1.37	No
CHAP bidirezionale ONTAP	1.27 - 1.37	No
Policy di esportazione dinamiche	1.27 - 1.37	No
Operatore Trident	1.27 - 1.37	No
Topologia CSI	1.27 - 1.37	No

Sistemi operativi host testati

Sebbene Trident non supporti ufficialmente sistemi operativi specifici, è noto che i seguenti sistemi funzionano:

- Versioni di Red Hat Enterprise Linux CoreOS (RHCOS) supportate da OpenShift Container Platform su AMD64 e ARM64
- Red Hat Enterprise Linux (RHEL) 8 o versione successiva su AMD64 e ARM64

  NVMe/TCP richiede RHEL 9 o versione successiva.

- Ubuntu 22.04 LTS o successivo su AMD64 e ARM64
- Windows Server 2022
- SUSE Linux Enterprise Server (SLES) 15 o versione successiva

Per impostazione predefinita, Trident viene eseguito in un container e quindi viene eseguito su qualsiasi lavoratore Linux. Tuttavia, tali dipendenti devono essere in grado di montare i volumi forniti da Trident utilizzando il client NFS standard o l'iniziatore iSCSI, a seconda dei backend in uso.

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

Trident potrebbe richiedere modifiche a un sistema di storage prima che possa essere utilizzato da una configurazione backend.

"Configurare i backend"

Porte Trident

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

"Porte Trident"

Immagini container e corrispondenti versioni di Kubernetes

Per le installazioni con montaggio ad aria, l'elenco seguente è un riferimento alle immagini contenitore necessarie per installare Trident. Utilizzare il `tridentctl images` comando per verificare l'elenco delle immagini contenitore necessarie.

L'elenco di immagini di Trident 26.10 riportato di seguito include le immagini per Kubernetes 1.37.

Immagini del container necessarie per Trident 26.10.1

Versioni di Kubernetes	Immagine container
v1.27.0, v1.28.0, v1.29.0, v1.30.0, v1.31.0, v1.32.0, v1.33.0, v1.34.0, v1.35.0, v1.36.0, v1.37.0	• docker.io/netapp/trident:26.10.1 • docker.io/netapp/trident-autosupport:26.10 • docker.io/netapp/trident-operator:26.10.1 (opzionale) • registry.k8s.io/sig-storage/csi-attacher:v4.11.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.17.0 • registry.k8s.io/sig-storage/csi-provisioner:v6.3.0 • registry.k8s.io/sig-storage/csi-resizer:v2.1.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.6.0

Immagini del container necessarie per Trident 26.06.2

Versioni di Kubernetes	Immagine container
v1.27.0, v1.28.0, v1.29.0, v1.30.0, v1.31.0, v1.32.0, v1.33.0, v1.34.0, v1.35.0, v1.36.0	• docker.io/netapp/trident:26.06.2 • docker.io/netapp/trident-autosupport:26.06 • docker.io/netapp/trident-operator:26.06.2 (opzionale) • registry.k8s.io/sig-storage/csi-attacher:v4.11.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.18.0 • registry.k8s.io/sig-storage/csi-provisioner:v6.3.0 • registry.k8s.io/sig-storage/csi-resizer:v2.1.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.6.0

Installare Trident

Informazioni sull'installazione di Trident

Per garantire che Trident possa essere installato in un'ampia varietà di ambienti e organizzazioni, NetApp offre diverse opzioni di installazione. È possibile installare Trident utilizzando l'operatore Trident (manualmente o utilizzando Helm) o con `tridentctl`. In questo argomento vengono fornite informazioni importanti per la scelta del processo di installazione appropriato.

Informazioni critiche su Trident 26.10

È necessario leggere le seguenti informazioni critiche su Trident.

**▼ informazioni aggiornate su Trident **

- Kubernetes 1.37 è ora supportato in Trident. Aggiorna Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipath 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.

Prima di iniziare

Indipendentemente dal percorso di installazione, è necessario disporre di:

- Privilegi completi per un cluster Kubernetes supportato che esegue una versione supportata di Kubernetes e con i requisiti delle funzionalità abilitati. Consultare "requisiti" per i dettagli.
- Accesso a un sistema storage NetApp supportato.
- Possibilità di montare volumi da tutti i nodi di lavoro Kubernetes.
- Un host Linux con `kubectl` (o. `oc`, Se si utilizza OpenShift) installato e configurato per gestire il cluster Kubernetes che si desidera utilizzare.
- Il `KUBECONFIG` Variabile d'ambiente impostata per puntare alla configurazione del cluster Kubernetes.
- Se utilizzi Kubernetes con Docker Enterprise, "Seguire la procedura per abilitare l'accesso CLI".
- Il cluster deve supportare workload con privilegi.



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

Scegliere il metodo di installazione desiderato

Seleziona il metodo di installazione più adatto alle tue esigenze. Dovresti anche esaminare le considerazioni per "passaggio da un metodo all'altro" prima di prendere una decisione.

Utilizzando l'operatore Trident

Che si utilizzi la distribuzione manuale o Helm, l'operatore Trident è un ottimo modo per semplificare l'installazione e gestire dinamicamente le risorse Trident. Puoi anche "Personalizzare l'implementazione dell'operatore Trident" utilizzando gli attributi nella `TridentOrchestrator` custom resource (CR).

I vantaggi derivanti dall'utilizzo dell'operatore Trident includono:

▼ ** Trident oggetto creag **

L'operatore Trident crea automaticamente i seguenti oggetti per la versione di Kubernetes.

- ServiceAccount per l'operatore
- ClusterRole e ClusterRoleBinding al ServiceAccount
- PodSecurityPolicy dedicata (per Kubernetes 1.25 e versioni precedenti)
- L'operatore stesso

▼ ** accountability**

L'operatore Trident cluster-scoped gestisce le risorse associate a un'installazione Trident a livello di cluster. In questo modo è possibile ridurre gli errori che potrebbero verificarsi quando si mantengono le risorse raggruppate nell'ambito del cluster utilizzando un operatore con ambito dello spazio dei nomi. Ciò è essenziale per auto-healing e patching.

▼ ** capabilitàdi **

L'operatore monitora l'installazione di Trident e adotta attivamente misure per risolvere problemi, ad esempio quando l'implementazione viene eliminata o se viene modificata accidentalmente. Viene creato un `trident-operator-<generated-id>` pod che associa un `TridentOrchestrator` CR a un'installazione Trident. Ciò garantisce che nel cluster sia presente una sola istanza di Trident e ne controlla la configurazione, assicurandosi che l'installazione sia idempotent. Quando vengono apportate modifiche all'installazione (ad esempio, l'eliminazione dell'implementazione o del demonset di nodi), l'operatore li identifica e li corregge singolarmente.

▼ ** Easy aggiorna l'installazione esistente **

È possibile aggiornare facilmente un'implementazione esistente con l'operatore. È sufficiente modificare `TridentOrchestrator` CR per aggiornare un'installazione.

Ad esempio, si consideri uno scenario in cui è necessario attivare Trident per generare registri di debug. Per fare questo, applicare una patch `TridentOrchestrator` al per impostare `spec.debug` su `true`:

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

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

▼ ** ha installato un tag **

L'operatore Trident, definito dall'ambito del cluster, consente la rimozione definitiva delle risorse definite dall'ambito del cluster. Gli utenti possono disinstallare completamente Trident e reinstallarlo facilmente.

▼ ** aggiornamento handlate **

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

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

Utilizzo di `tridentctl`

Se si dispone di una distribuzione esistente che deve essere aggiornata o si desidera personalizzare notevolmente la distribuzione, è opportuno prendere in considerazione . Questo è il metodo convenzionale di distribuzione di Trident.

È possibile generare i manifesti per le risorse Trident. Sono inclusi la distribuzione, il daemesort, l'account di servizio e il ruolo cluster creato da Trident durante l'installazione.

A partire dalla release 22,04, le chiavi AES non verranno più rigenerate ogni volta che viene installato Trident. Con questa versione, Trident installerà un nuovo oggetto segreto che persiste nelle installazioni. Ciò significa che `tridentctl` in 22,04 è possibile disinstallare le versioni precedenti di Trident, ma le versioni precedenti non possono disinstallare le installazioni 22,04. Selezionare il *metodo* di installazione appropriato.

Scegliere la modalità di installazione

Determinare il processo di implementazione in base alla *modalità di installazione* (Standard, Offline o Remote) richiesta dall'organizzazione.

Installazione standard

Questo è il modo più semplice per installare Trident e funziona per la maggior parte degli ambienti che non impongono restrizioni di rete. La modalità di installazione standard utilizza registri predefiniti per memorizzare le (`registry.k8s.io`immagini Trident (`docker.io`) e CSI) richieste.

Quando si utilizza la modalità standard, il programma di installazione di Trident:

- Recupera le immagini container su Internet
- Crea un'implementazione o un daemonset di nodi, che accelera i pod Trident su tutti i nodi idonei del cluster Kubernetes

Installazione offline

La modalità di installazione offline potrebbe essere richiesta in un luogo sicuro o con aria compressa. In questo scenario, è possibile creare un singolo registro privato mirrorato o due registri mirrorati per memorizzare le immagini Trident e CSI richieste.

Indipendentemente dalla configurazione del Registro di sistema, le immagini CSI devono risiedere in un unico Registro di sistema.

Installazione remota

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

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

Selezionare il processo in base al metodo e alla modalità

Dopo aver preso le decisioni, selezionare il processo appropriato.

Metodo	Modalità di installazione
Operatore Trident (manualmente)	"Installazione standard" "Installazione offline"
Operatore Trident (Helm)	"Installazione standard" "Installazione offline"
`tridentctl`	"Installazione standard o offline"

Passaggio da un metodo di installazione all'altro

È possibile modificare il metodo di installazione. Prima di procedere, considerare quanto segue:

- Utilizzare sempre lo stesso metodo per installare e disinstallare Trident. Se è stato distribuito con `tridentctl`, è necessario utilizzare la versione appropriata del `tridentctl` file binario per disinstallare Trident. Allo stesso modo, se si esegue la distribuzione con l'operatore, è necessario modificare il `TridentOrchestrator` CR e impostare la `spec.uninstall=true` disinstallazione di Trident.
- Se si dispone di una distribuzione basata su operatore che si desidera rimuovere e utilizzare invece `tridentctl` per distribuire Trident, è necessario prima modificare `TridentOrchestrator` e impostare la `spec.uninstall=true` disinstallazione di Trident. Quindi eliminare `TridentOrchestrator` e l'implementazione dell'operatore. È quindi possibile installare utilizzando `tridentctl`.
- Se si dispone di una distribuzione manuale basata sull'operatore e si desidera utilizzare la distribuzione dell'operatore Trident basata su Helm, è necessario disinstallare prima manualmente l'operatore e quindi eseguire l'installazione di Helm. Ciò consente a Helm di distribuire l'operatore Trident con le etichette e le annotazioni richieste. In caso contrario, la distribuzione dell'operatore Trident basato su Helm non riuscirà, con errori di convalida dell'etichetta e di convalida dell'annotazione.
- Se hai un `tridentctl` distribuzione basata su, è possibile eseguire la distribuzione basata su Helm o su Operator senza disinstallare Trident.

Altre opzioni di configurazione note

Quando si installa Trident sui prodotti del portfolio VMware Tanzu:

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

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

Installare utilizzando l'operatore Trident

Implementare manualmente l'operatore Trident (modalità standard)

È possibile distribuire manualmente l'operatore Trident per installare Trident. Questo processo si applica alle installazioni in cui le immagini contenitore richieste da Trident non sono memorizzate in un registro privato. Se si dispone di un registro delle immagini privato, utilizzare "processo per l'implementazione offline".

Informazioni critiche su Trident 26.10

È necessario leggere le seguenti informazioni critiche su Trident.

**▼ informazioni aggiornate su Trident **

- Trident 26.10 supporta Kubernetes dalla versione 1.27 alla 1.37. Controlla l'elenco delle immagini specifiche per la versione in "Requisiti Trident" prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipath 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.

Implementare manualmente l'operatore Trident e installare Trident

Verificate "panoramica dell'installazione" per assicurarvi di aver soddisfatto i prerequisiti di installazione e di aver selezionato l'opzione di installazione corretta per il vostro ambiente.

Prima di iniziare

Prima di iniziare l'installazione, accedi all'host Linux e verifica che stia gestendo un "Cluster Kubernetes supportato" funzionante e che tu disponga dei privilegi necessari.

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

▼ Details

1. Verificare la versione di Kubernetes:

```
kubectl version
```

2. Verificare i privilegi di amministratore del cluster:

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

3. Verificare che sia possibile avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il sistema di storage tramite la rete pod:

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

Fase 1: Scaricare il pacchetto di installazione di Trident

Il pacchetto di installazione di Trident contiene tutto ciò che serve per distribuire l'operatore Trident e installare Trident. Scaricare ed estrarre la versione più recente del programma di installazione di Trident da "La sezione *Assets* su GitHub".

```
wget https://github.com/NetApp/trident/releases/download/v26.10.0/trident-
installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.tar.gz
cd trident-installer
```

Fase 2: Creare `TridentOrchestrator` CRD

Crea il `TridentOrchestrator` Definizione di risorse personalizzate (CRD). Creerai un `TridentOrchestrator` Risorsa personalizzata in seguito. Utilizzare la versione CRD YAML appropriata in `deploy/crds` per creare il `TridentOrchestrator` CRD.

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

Fase 3: Implementare l'operatore Trident

Il programma di installazione di Trident fornisce un file bundle che può essere utilizzato per installare l'operatore e creare oggetti associati. Il file bundle è un modo semplice per distribuire l'operatore e installare Trident utilizzando una configurazione predefinita.

- Per i cluster che eseguono Kubernetes 1,24, 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

- Per impostazione predefinita, il programma di installazione di Trident implementa l'operatore in `trident` namespace. Se il `trident` namespace inesistente, crearlo utilizzando:

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

- Per implementare l'operatore in uno spazio dei nomi diverso da `trident` namespace, aggiornamento `serviceaccount.yaml`, `clusterrolebinding.yaml` e. `operator.yaml` e generare il file bundle utilizzando `kustomization.yaml`.

  a. Creare il `kustomization.yaml` usando il seguente comando dove *<bundle.yaml>* è `bundle_pre_1_25.yaml` oppure `bundle_post_1_25.yaml` In base alla versione di Kubernetes.

  ```
  cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
  ```

  b. Compilate il pacchetto usando il seguente comando dove *<bundle.yaml>* è `bundle_pre_1_25.yaml` oppure `bundle_post_1_25.yaml` In base alla versione di Kubernetes.

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

Fasi

1. Creare le risorse e implementare l'operatore:

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

2. Verificare che l'operatore, l'implementazione e i replicaset siano stati creati.

   ```
   kubectl get all -n <operator-namespace>
   ```

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

Fase 4: Creare `TridentOrchestrator` E installare Trident

A questo punto è possibile creare `TridentOrchestrator` e installare Trident. Facoltativamente, è possibile "Personalizzare l'installazione di Trident" utilizzare gli attributi nella `TridentOrchestrator` specifica.

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

kubectl describe torc trident

Name:         trident
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
...
```

```
Spec:
  Debug:     true
  Namespace: trident
  nodePrep:
  - iscsi
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:26.10
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:26.10.1
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v26.10.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Verificare l'installazione

Esistono diversi modi per verificare l'installazione.

Utilizzo di `TridentOrchestrator` stato

Lo stato di `TridentOrchestrator` Indica se l'installazione ha avuto esito positivo e visualizza la versione di Trident installata. Durante l'installazione, lo stato di `TridentOrchestrator` modifiche da `Installing` a. `Installed`. Se si osserva `Failed` e l'operatore non è in grado di ripristinarsi da solo, "controllare i registri".

Stato	Descrizione
Installazione in corso	L'operatore sta installando Trident utilizzando questo `TridentOrchestrator` CR.
Installato	Installazione di Trident riuscita.
Disinstallazione in corso	L'operatore sta disinstallando Trident, perché `spec.uninstall=true`.

Stato	Descrizione
Disinstallato	Trident è disinstallato.
Non riuscito	L'operatore non ha potuto installare, applicare patch, aggiornare o disinstallare Trident; l'operatore tenterà automaticamente di ripristinare da questo stato. Se lo stato persiste, è necessario eseguire la risoluzione dei problemi.
Aggiornamento in corso	L'operatore sta aggiornando un'installazione esistente.
Errore	Il `TridentOrchestrator` non viene utilizzato. Un'altra esiste già.

Utilizzo dello stato di creazione del pod

È possibile verificare se l'installazione di Trident è stata completata controllando lo stato dei pod creati:

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw        6/6     Running   0
1m
trident-node-linux-mr6zc                   2/2     Running   0
1m
trident-node-linux-xrp7w                   2/2     Running   0
1m
trident-node-linux-zh2jt                   2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv          1/1     Running   0
3m
```

Utilizzo di `tridentctl`

È possibile utilizzare `tridentctl` per controllare la versione di Trident installata.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.10.1        | 26.10.1        |
+----------------+----------------+
```

Implementare manualmente l'operatore Trident (modalità offline)

È possibile distribuire manualmente l'operatore Trident per installare Trident. Questo processo si applica alle installazioni in cui le immagini contenitore richieste da Trident sono memorizzate in un registro privato. Se non si dispone di un registro delle immagini private, utilizzare "processo per l'implementazione standard".

Informazioni critiche su Trident

È necessario leggere le seguenti informazioni critiche su Trident.

▼ ** informazioni aggiornate su Trident **

- Trident 26.10 supporta Kubernetes dalla versione 1.27 alla 1.37. Controlla l'elenco delle immagini specifiche per la versione in "Requisiti Trident" prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipath 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.

Implementare manualmente l'operatore Trident e installare Trident

Verificate "panoramica dell'installazione" per assicurarvi di aver soddisfatto i prerequisiti di installazione e di aver selezionato l'opzione di installazione corretta per il vostro ambiente.

Prima di iniziare

Accedi all'host Linux e verifica che stia gestendo un sistema operativo funzionante "Cluster Kubernetes supportato" e che tu disponga dei privilegi necessari.

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

▼ **Details**

1. Verificare la versione di Kubernetes:

```
kubectl version
```

2. Verificare i privilegi di amministratore del cluster:

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

3. Verificare che sia possibile avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il sistema di storage tramite la rete pod:

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

Fase 1: Scaricare il pacchetto di installazione di Trident

Il pacchetto di installazione di Trident contiene tutto ciò che serve per distribuire l'operatore Trident e installare Trident. Scaricare ed estrarre la versione più recente del programma di installazione di Trident da "La sezione *Assets* su GitHub".

```
wget https://github.com/NetApp/trident/releases/download/v26.10.0/trident-installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.tar.gz
cd trident-installer
```

Fase 2: Creare `TridentOrchestrator` CRD

Crea il `TridentOrchestrator` Definizione di risorse personalizzate (CRD). Creerai un `TridentOrchestrator` Risorse personalizzate in seguito. Utilizzare la versione CRD YAML appropriata in `deploy/crds` per creare il `TridentOrchestrator` CRD:

```
kubectl create -f deploy/crds/<VERSION>.yaml
```

Fase 3: Aggiornare la posizione del registro nell'operatore

In `/deploy/operator.yaml`, aggiorna `image: docker.io/netapp/trident-operator:26.10.1` per riflettere la posizione del tuo registro delle immagini. Le tue "Immagini Trident e CSI" possono essere localizzate in un unico registro o in registri diversi, ma tutte le immagini CSI devono trovarsi nello stesso registro. Ad esempio:

- `image: <your-registry>/trident-operator:26.10.1` se tutte le immagini si trovano in un unico registro.
- `image: <your-registry>/netapp/trident-operator:26.10.1` se la tua immagine Trident si trova in un registro diverso rispetto alle tue immagini CSI.

Fase 4: Implementare l'operatore Trident

Il programma di installazione di Trident fornisce un file bundle che può essere utilizzato per installare l'operatore e creare oggetti associati. Il file bundle è un modo semplice per distribuire l'operatore e installare Trident utilizzando una configurazione predefinita.

- Per i cluster che eseguono Kubernetes 1,24, 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

- Per impostazione predefinita, il programma di installazione di Trident implementa l'operatore in `trident` namespace. Se il `trident` namespace inesistente, crearlo utilizzando:

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

- Per implementare l'operatore in uno spazio dei nomi diverso da `trident` namespace, aggiornamento `serviceaccount.yaml`, `clusterrolebinding.yaml` e. `operator.yaml` e generare il file bundle utilizzando `kustomization.yaml`.

  a. Creare il `kustomization.yaml` usando il seguente comando dove *<bundle.yaml>* è `bundle_pre_1_25.yaml` oppure `bundle_post_1_25.yaml` In base alla versione di Kubernetes.

  ```
  cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
  ```

  b. Compilate il pacchetto usando il seguente comando dove *<bundle.yaml>* è `bundle_pre_1_25.yaml` oppure `bundle_post_1_25.yaml` In base alla versione di Kubernetes.

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

Fasi

1. Creare le risorse e implementare l'operatore:

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

2. Verificare che l'operatore, l'implementazione e i replicaset siano stati creati.

```
kubectl get all -n <operator-namespace>
```

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

Fase 5: Aggiornare la posizione del registro delle immagini in `TridentOrchestrator`

Il tuo "Immagini Trident e CSI" Può trovarsi in un registro o in registri diversi, ma tutte le immagini CSI devono trovarsi nello stesso registro. Aggiornare `deploy/crds/tridentorchestrator_cr.yaml` per aggiungere le specifiche di posizione aggiuntive in base alla configurazione del registro di sistema.

Immagini in un registro

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.10"
tridentImage: "<your-registry>/trident:26.10.1"
```

Immagini in diversi registri

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.10"
tridentImage: "<your-registry>/trident:26.10.1"
```

Fase 6: Creare `TridentOrchestrator` E installare Trident

A questo punto è possibile creare `TridentOrchestrator` e installare Trident. Facoltativamente, è possibile utilizzare ulteriormente "Personalizzare l'installazione di Trident"gli attributi nella `TridentOrchestrator` specifica. L'esempio seguente mostra un'installazione in cui le immagini Trident e CSI si trovano in diversi registri.

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

kubectl describe torc trident

Name:         trident
Namespace:
Labels:       <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Spec:
  Autosupport Image:  <your-registry>/trident-autosupport:26.10
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.10.1
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/trident-autosupport:26.10
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
```

```
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/trident:26.10.1
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v26.10.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Verificare l'installazione

Esistono diversi modi per verificare l'installazione.

Utilizzo di `TridentOrchestrator` stato

Lo stato di `TridentOrchestrator` Indica se l'installazione ha avuto esito positivo e visualizza la versione di Trident installata. Durante l'installazione, lo stato di `TridentOrchestrator` modifiche da `Installing` a. `Installed`. Se si osserva `Failed` e l'operatore non è in grado di ripristinarsi da solo, "controllare i registri".

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

Utilizzo dello stato di creazione del pod

È possibile verificare se l'installazione di Trident è stata completata controllando lo stato dei pod creati:

```
kubectl get pods -n trident
```

```
NAME                                   READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw    6/6     Running   0
1m
trident-node-linux-mr6zc               2/2     Running   0
1m
trident-node-linux-xrp7w               2/2     Running   0
1m
trident-node-linux-zh2jt               2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv      1/1     Running   0
3m
```

Utilizzo di `tridentctl`

È possibile utilizzare `tridentctl` per controllare la versione di Trident installata.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.10.1        | 26.10.1        |
+----------------+----------------+
```

Implementare l'operatore Trident utilizzando Helm (modalità standard)

È possibile distribuire l'operatore Trident e installare Trident utilizzando Helm. Questo processo si applica alle installazioni in cui le immagini contenitore richieste da Trident non sono memorizzate in un registro privato. Se si dispone di un registro delle immagini privato, utilizzare "processo per l'implementazione offline".

Informazioni critiche su Trident 26.10

È necessario leggere le seguenti informazioni critiche su Trident.

▼ ** informazioni aggiornate su Trident **

- Trident 26.10 supporta Kubernetes dalla versione 1.27 alla 1.37. Controlla l'elenco delle immagini specifiche per la versione in "Requisiti Trident" prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipath 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.

Distribuire l'operatore Trident e installare Trident utilizzando Helm

Utilizzo di Trident "Grafico di comando" È possibile implementare l'operatore Trident e installare Trident in un'unica fase.

Verificate "panoramica dell'installazione" per assicurarvi di aver soddisfatto i prerequisiti di installazione e di aver selezionato l'opzione di installazione corretta per il vostro ambiente.

Prima di iniziare

Oltre al "prerequisiti per l'implementazione" è necessario "Helm versione 3".

Fasi

1. Aggiungere il repository Trident Helm:

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

2. Usa `helm install` e specifica un nome per la distribuzione come nel seguente esempio in cui `100.2610.0` è la versione di Trident che stai installando.

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
--create-namespace --namespace <trident-namespace>
```

> Se è già stato creato uno spazio dei nomi per Trident, il `--create-namespace` il parametro non crea uno spazio dei nomi aggiuntivo.

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.

Passare i dati di configurazione durante l'installazione

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

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

Ad esempio, per modificare il valore predefinito di `debug`, eseguire il seguente comando dove `100.2610.0` è la versione di Trident che si sta installando:

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
--create-namespace --namespace trident --set tridentDebug=true
```

Opzioni di configurazione

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

Opzione	Descrizione	Predefinito
`nodeSelector`	Etichette dei nodi per l'assegnazione dei pod	
`podAnnotations`	Annotazioni Pod	
`deploymentAnnotations`	Annotazioni di implementazione	
`tolerations`	Pedaggi per l'assegnazione del pod	

`affinity`	Affinità per l'assegnazione del pod	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedulingIgnoredDuringExecution:`
		`  nodeSelectorTerms:`
		`  - matchExpressions:`
		`  - key: kubernetes.io/arch`
		`  operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key: kubernetes.io/os`
		`  operator: In`
		`  values:`
		`  - linux`
		
		Non rimuovere l'affinità predefinita dal file Values.yaml. Se si desidera fornire un'affinità personalizzata, estendere l'affinità predefinita.

`tridentControllerPluginNodeSelector`	Selettori di nodi aggiuntivi per i pod. Fare riferimento a. Comprensione dei pod controller e dei pod di nodi per ulteriori informazioni.	

Opzione	Descrizione	Predefinito
`tridentControllerPluginTolerations`	Ignora le tolleranze Kubernetes per i pod. Fare riferimento a. Comprensione dei pod controller e dei pod di nodi per ulteriori informazioni.	
`tridentNodePluginNodeSelector`	Selettori di nodi aggiuntivi per i pod. Fare riferimento a. Comprensione dei pod controller e dei pod di nodi per ulteriori informazioni.	
`tridentNodePluginTolerations`	Ignora le tolleranze Kubernetes per i pod. Fare riferimento a. Comprensione dei pod controller e dei pod di nodi per ulteriori informazioni.	
`imageRegistry`	Identifica il Registro di sistema per `trident-operator` , `trident` e altre immagini. Lasciare vuoto per accettare l'impostazione predefinita. IMPORTANTE: Quando si installa Trident in un repository privato, se si utilizza `imageRegistry` lo switch per specificare la posizione del repository, non utilizzare `/netapp/` nel percorso del repository.	`""`
`imagePullPolicy`	Imposta il criterio di pull dell'immagine per `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Imposta i segreti di pull dell'immagine per `trident-operator`, `trident`e altre immagini.	
`kubeletDir`	Consente di ignorare la posizione host dello stato interno del kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Consente di impostare il livello di log dell'operatore Trident su: `trace`, `debug`, `info`, `warn`, `error`, o. `fatal`.	`"info"`
`operatorDebug`	Consente di impostare il livello di log dell'operatore Trident su debug.	`true`
`operatorImage`	Consente di eseguire l'override completo dell'immagine per `trident-operator`.	`""`
`operatorImageTag`	Consente di sovrascrivere il tag di `trident-operator` immagine.	`""`

Opzione	Descrizione	Predefinito
`tridentIPv6`	Permette a Trident di lavorare in cluster IPv6.	`false`
`tridentK8sTimeout`	Esegue l'override del timeout predefinito di 30 secondi per la maggior parte delle operazioni API di Kubernetes (se diverso da zero, in secondi).	`0`
`tridentHttpRequestTimeout`	Esegue l'override del timeout predefinito di 90 secondi per le richieste HTTP, con `0s` è una durata infinita per il timeout. Non sono consentiti valori negativi.	`"90s"`
`tridentSilenceAutosupport`	Consente di disattivare la creazione di rapporti periodici AutoSupport Trident.	`false`
`tridentExcludeAutosupport`	Quando è impostato su `true`, Trident non configura il container AutoSupport durante l'installazione.	`false`
`tridentAutosupportImageTag`	Consente di sovrascrivere il tag dell'immagine per il contenitore Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Consente al contenitore Trident AutoSupport di telefonare a casa tramite un proxy HTTP.	`""`
`tridentLogFormat`	Consente di impostare il formato di registrazione Trident (`text`o `json`).	`"text"`
`tridentDisableAuditLog`	Disattiva il registro di controllo Trident.	`true`
`tridentLogLevel`	Consente di impostare il livello di registrazione di Trident su: `trace`, `debug`, `info`, , `warn` `error` O `fatal`.	`"info"`
`tridentDebug`	Consente di impostare il livello di registro di Trident su `debug` . È possibile automatizzare il processo di distacco forzato tramite l'integrazione con l'operatore di controllo dello stato del nodo (NHC). Per informazioni, vedere"Automazione del failover delle applicazioni stateful con Trident" .	`false`

Opzione	Descrizione	Predefinito
`tridentLogWorkflows`	Consente di attivare flussi di lavoro Trident specifici per la registrazione delle tracce o l'eliminazione dei registri.	`""`
`tridentLogLayers`	Consente di attivare livelli Trident specifici per la registrazione delle tracce o la soppressione dei registri.	`""`
`tridentImage`	Consente la sostituzione completa dell'immagine per Trident.	`""`
`tridentImageTag`	Consente di sovrascrivere il tag dell'immagine per Trident.	`""`
`tridentProbePort`	Consente di ignorare la porta predefinita utilizzata per le sonde liveness/readiness Kubernetes.	`""`
`windows`	Consente di installare Trident sul nodo di lavoro Windows.	`false`
`enableForceDetach`	Consente di attivare la funzione di distacco forzato.	`false`
`enableDataLIFRefresh`	Aggiorna i LIF dati NVMe per i volumi pubblicati quando i LIF vengono aggiunti, rimossi o assegnati a un nuovo indirizzo. Si applica a `ontap-san` con `sanType: nvme` su ONTAP unificato e NVMe/TCP su ASA r2. Per ulteriori informazioni, vedi "Aggiorna i LIF dei dati NVMe".	`false`
`excludePodSecurityPolicy`	Esclude la creazione della policy di sicurezza del pod operatore.	`false`
`cloudProvider`	Impostato su `"Azure"` quando utilizzi identità gestite o un'identità cloud su un cluster AKS, `"AWS"` quando utilizzi un'identità cloud su un cluster EKS o `"GCP"` quando utilizzi un'identità cloud su GKE o Workload Identity Federation su OpenShift.	`""`

Opzione	Descrizione	Predefinito
`cloudIdentity`	Imposta l'identità del carico di lavoro ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx") quando usi l'identità cloud su un cluster AKS. Imposta il ruolo AWS IAM ("'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role'") quando usi l'identità cloud su un cluster EKS. Imposta l'annotazione dell'identità cloud GKE quando usi l'identità cloud su GKE. Non impostare per la federazione dell'identità del carico di lavoro di OpenShift.	`""`
`gcpWIFAudience`	Audience per il token dell'account di servizio proiettato utilizzato con Google Cloud Workload Identity Federation su OpenShift. Usa il nome canonico della risorsa del provider. Per ulteriori informazioni, consulta "Configura Google Cloud NetApp Volumes con Workload Identity Federation".	`""`
`iscsiSelfHealingInterval`	Intervallo in cui viene richiamata la riparazione automatica iSCSI.	`5m0s`
`iscsiSelfHealingWaitTime`	La durata dopo la quale l'autoriparazione iSCSI avvia un tentativo di risolvere una sessione obsoleta eseguendo una disconnessione e un login successivo.	`7m0s`
`nodePrep`	Consente a Trident di preparare i nodi del cluster Kubernetes per gestire i volumi utilizzando il protocollo di archiviazione dati specificato. **Attualmente, `iscsi` è l'unico valore supportato.** NOTA: a partire da OpenShift 4.19, la versione minima Trident supportata per questa funzionalità è 25.06.1.	
`enableConcurrency`	Consente operazioni simultanee del controller Trident per un throughput migliore. Per ulteriori informazioni sui driver supportati, vedi il xref:./trident-install/ Controller Scalability documentation.	falso

Opzione	Descrizione	Predefinito
`k8sAPIQPS`	Limite di query al secondo (QPS) utilizzato dal controller durante la comunicazione con il server API Kubernetes. Il valore Burst viene impostato automaticamente in base al valore QPS.	`100`; facoltativo

Opzione	Descrizione	Predefinito

`resources`	Imposta i limiti delle risorse Kubernetes e le richieste per i pod del controller, del nodo e dell'operatore Trident . È possibile configurare CPU e memoria per ogni contenitore e sidecar per gestire l'allocazione delle risorse in Kubernetes.
	
	Per ulteriori informazioni sulla configurazione delle richieste e dei limiti delle risorse, fare riferimento a"Gestione delle risorse per pod e container" .
	
	• NON modificare i nomi di alcun contenitore o campo.
	• NON modificare l'indentazione: l'indentazione YAML è fondamentale per una corretta analisi.
	
	• Per impostazione predefinita non vengono applicati limiti: solo le richieste hanno valori predefiniti e vengono applicate automaticamente se non specificate.
	• I nomi dei contenitori sono elencati così come appaiono nelle specifiche del pod.
	• I sidecar sono elencati sotto ogni contenitore principale.
	
	Controlla i TORC
		<pre>resources:
		  controller:
		    trident-main:
		      requests:
		        cpu: 10m
		        memory: 80Mi
		      limits:
		          cpu:
		          memory:
		    csi-provisioner:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-attacher:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-resizer:
		      requests:
		        cpu: 3m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-snapshotter:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    trident-autosupport:
		      requests:
		        cpu: 1m
		        memory: 30Mi
		      limits:
		        cpu:
		        memory:
		  node:</pre>

Opzione	Descrizione	Predefinito
`httpsMetrics`	Abilita HTTPS per l'endpoint delle metriche Prometheus.	falso
`hostNetwork`	Abilita la rete host per il controller Trident . Questa funzionalità è utile quando si desidera separare il traffico frontend da quello backend in una rete multi-home.	falso

Comprensione dei pod controller e dei pod di nodi

Trident funziona come un singolo pod controller, più un pod di nodo su ciascun nodo di lavoro nel cluster. Il pod di nodo deve essere in esecuzione su qualsiasi host in cui si desidera montare potenzialmente un volume Trident.

Kubernetes "selettori di nodi" e. "tollerazioni e contamini" vengono utilizzati per vincolare l'esecuzione di un pod su un nodo specifico o preferito. Utilizzo di`ControllerPlugin` e. `NodePlugin`, è possibile specificare vincoli e override.

- 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.

Implementare l'operatore Trident utilizzando Helm (modalità offline)

È possibile distribuire l'operatore Trident e installare Trident utilizzando Helm. Questo processo si applica alle installazioni in cui le immagini contenitore richieste da Trident sono memorizzate in un registro privato. Se non si dispone di un registro delle immagini private, utilizzare "processo per l'implementazione standard".

Informazioni critiche su Trident 26.10

È necessario leggere le seguenti informazioni critiche su Trident.

**▼ informazioni aggiornate su Trident **

- Trident 26.10 supporta Kubernetes dalla versione 1.27 alla 1.37. Controlla l'elenco delle immagini specifiche per la versione in "Requisiti Trident" prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipath 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.

Distribuire l'operatore Trident e installare Trident utilizzando Helm

Utilizzo di Trident "Grafico di comando" È possibile implementare l'operatore Trident e installare Trident in un'unica fase.

Verificate "panoramica dell'installazione" per assicurarvi di aver soddisfatto i prerequisiti di installazione e di aver selezionato l'opzione di installazione corretta per il vostro ambiente.

Prima di iniziare

Oltre al "prerequisiti per l'implementazione" è necessario "Helm versione 3".

> Quando si installa Trident in un repository privato, se si utilizza `imageRegistry` lo switch per specificare la posizione del repository, non utilizzare `/netapp/` nel percorso del repository.

Fasi

1. Aggiungere il repository Trident Helm:

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

2. Usa `helm install` e specifica un nome per il deployment e la posizione del registro delle immagini. Le tue "Immagini Trident e CSI" possono essere localizzate in un unico registro o in registri diversi, ma tutte le immagini CSI devono trovarsi nello stesso registro. Negli esempi, `100.2610.0` è la versione di Trident che stai installando.

Immagini in un registro

```
helm install <name> netapp-trident/trident-operator --version
100.2610.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace> --set nodePrep={iscsi}
```

Immagini in diversi registri

```
helm install <name> netapp-trident/trident-operator --version
100.2610.0 --set imageRegistry=<your-registry> --set operatorImage
=<your-registry>/trident-operator:26.10.1 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:26.10
--set tridentImage=<your-registry>/trident:26.10.1 --create
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```

> Se è già stato creato uno spazio dei nomi per Trident, il `--create-namespace` il parametro non crea uno spazio dei nomi aggiuntivo.

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.

Passare i dati di configurazione durante l'installazione

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

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

Ad esempio, per modificare il valore predefinito di `debug`, eseguire il seguente comando dove `100.2610.0` è la versione di Trident che si sta installando:

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
--create-namespace --namespace trident --set tridentDebug=true
```

Per aggiungere il valore nodePrep, eseguire il comando seguente:

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
--create-namespace --namespace trident --set nodePrep={iscsi}
```

Opzioni di configurazione

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

> Non rimuovere l'affinità predefinita dal file Values.yaml. Se si desidera fornire un'affinità personalizzata, estendere l'affinità predefinita.

Opzione	Descrizione	Predefinito
`nodeSelector`	Etichette dei nodi per l'assegnazione dei pod	
`podAnnotations`	Annotazioni Pod	
`deploymentAnnotations`	Annotazioni di implementazione	
`tolerations`	Pedaggi per l'assegnazione del pod	

Opzione	Descrizione	Predefinito

`affinity`	Affinità per l'assegnazione del pod	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedul`
		`ingIgnoredDuringExecu`
		`tion:`
		
		`nodeSelectorTerms:`
		`  -`
		`matchExpressions:`
		`  - key:`
		`kubernetes.io/arch`
		
		`operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key:`
		`kubernetes.io/os`
		
		`operator: In`
		`  values:`
		`  - linux`
		
		Non rimuovere l'affinità predefinita dal file Values.yaml. Se si desidera fornire un'affinità personalizzata, estendere l'affinità predefinita.

`tridentControllerPluginNodeSelector`	Selettori di nodi aggiuntivi per i pod. Fare riferimento a. "Comprensione dei pod controller e dei pod di nodi" per ulteriori informazioni.	
`tridentControllerPluginTolerations`	Ignora le tolleranze Kubernetes per i pod. Fare riferimento a. "Comprensione dei pod controller e dei pod di nodi" per ulteriori informazioni.	

Opzione	Descrizione	Predefinito
`tridentNodePluginNodeSelector`	Selettori di nodi aggiuntivi per i pod. Fare riferimento a. "Comprensione dei pod controller e dei pod di nodi" per ulteriori informazioni.	
`tridentNodePluginTolerations`	Ignora le tolleranze Kubernetes per i pod. Fare riferimento a. "Comprensione dei pod controller e dei pod di nodi" per ulteriori informazioni.	
`imageRegistry`	Identifica il Registro di sistema per `trident-operator` , `trident` e altre immagini. Lasciare vuoto per accettare l'impostazione predefinita. IMPORTANTE: Quando si installa Trident in un repository privato, se si utilizza `imageRegistry` lo switch per specificare la posizione del repository, non utilizzare `/netapp/` nel percorso del repository.	""
`imagePullPolicy`	Imposta il criterio di pull dell'immagine per `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Imposta i segreti di pull dell'immagine per `trident-operator`, `trident`e altre immagini.	
`kubeletDir`	Consente di ignorare la posizione host dello stato interno del kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Consente di impostare il livello di log dell'operatore Trident su: `trace`, `debug`, `info`, `warn`, `error`, o. `fatal`.	`"info"`
`operatorDebug`	Consente di impostare il livello di log dell'operatore Trident su debug.	`true`
`operatorImage`	Consente di eseguire l'override completo dell'immagine per `trident-operator`.	""
`operatorImageTag`	Consente di sovrascrivere il tag di `trident-operator` immagine.	""
`tridentIPv6`	Permette a Trident di lavorare in cluster IPv6.	`false`

Opzione	Descrizione	Predefinito
`tridentK8sTimeout`	Sostituisce il timeout predefinito di 180 secondi per la maggior parte delle operazioni API di Kubernetes (se diverso da zero, in secondi). IL `tridentK8sTimeout` il parametro è applicabile solo per l'installazione Trident .	`180`
`tridentHttpRequestTimeout`	Esegue l'override del timeout predefinito di 90 secondi per le richieste HTTP, con `0s` è una durata infinita per il timeout. Non sono consentiti valori negativi.	`"90s"`
`tridentSilenceAutosupport`	Consente di disattivare la creazione di rapporti periodici AutoSupport Trident.	`false`
`tridentExcludeAutosupport`	Quando è impostato su `true`, Trident non configura il container AutoSupport durante l'installazione.	`false`
`tridentAutosupportImageTag`	Consente di sovrascrivere il tag dell'immagine per il contenitore Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Consente al contenitore Trident AutoSupport di telefonare a casa tramite un proxy HTTP.	""
`tridentLogFormat`	Consente di impostare il formato di registrazione Trident (`text`o `json`).	`"text"`
`tridentDisableAuditLog`	Disattiva il registro di controllo Trident.	`true`
`tridentLogLevel`	Consente di impostare il livello di registrazione di Trident su: `trace`, `debug`, `info`, , `warn` `error` O `fatal`.	`"info"`
`tridentDebug`	Consente di impostare il livello di registro di Trident su `debug`.	`false`
`tridentLogWorkflows`	Consente di attivare flussi di lavoro Trident specifici per la registrazione delle tracce o l'eliminazione dei registri.	""

Opzione	Descrizione	Predefinito
`tridentLogLayers`	Consente di attivare livelli Trident specifici per la registrazione delle tracce o la soppressione dei registri.	""
`tridentImage`	Consente la sostituzione completa dell'immagine per Trident.	""
`tridentImageTag`	Consente di sovrascrivere il tag dell'immagine per Trident.	""
`tridentProbePort`	Consente di ignorare la porta predefinita utilizzata per le sonde liveness/readiness Kubernetes.	""
`windows`	Consente di installare Trident sul nodo di lavoro Windows.	`false`
`enableForceDetach`	Consente di abilitare la funzione di distacco forzato. È possibile automatizzare il processo di distacco forzato tramite l'integrazione con l'operatore di controllo dello stato del nodo (NHC). Per informazioni, vedere"Automazione del failover delle applicazioni stateful con Trident" .	`false`
`enableDataLIFRefresh`	Aggiorna i LIF dati NVMe per i volumi pubblicati quando i LIF vengono aggiunti, rimossi o assegnati a un nuovo indirizzo. Si applica a `ontap-san` con `sanType: nvme` su ONTAP unificato e NVMe/TCP su ASA r2. Per ulteriori informazioni, vedi "Aggiorna i LIF dei dati NVMe".	`false`
`excludePodSecurityPolicy`	Esclude la creazione della policy di sicurezza del pod operatore.	`false`

Opzione	Descrizione	Predefinito
`nodePrep`	Permette a Trident di preparare i nodi del cluster Kubernetes per gestire i volumi utilizzando il protocollo storage specificato. **Attualmente, `iscsi` è l'unico valore supportato.** A partire da OpenShift 4.19, la versione minima Trident supportata per questa funzionalità è 25.06.1.	

Opzione	Descrizione	Predefinito

`resources`	Imposta i limiti delle risorse Kubernetes e le richieste per i pod del controller, del nodo e dell'operatore Trident . È possibile configurare CPU e memoria per ogni contenitore e sidecar per gestire l'allocazione delle risorse in Kubernetes.
	
	Per ulteriori informazioni sulla configurazione delle richieste e dei limiti delle risorse, fare riferimento a"Gestione delle risorse per pod e container" .
	
	• NON modificare i nomi di alcun contenitore o campo.
	• NON modificare l'indentazione: l'indentazione YAML è fondamentale per una corretta analisi.
	
	• Per impostazione predefinita non vengono applicati limiti: solo le richieste hanno valori predefiniti.
	• I nomi dei contenitori sono elencati così come appaiono nelle specifiche del pod.
	• I sidecar sono elencati sotto ogni contenitore principale.
	• Controlla i TORC `status.CurrentInstallationParams` campo per visualizzare i
		`resources:`
		`  controller:`
		`  trident-main:`
		`  requests:`
		`  cpu: 10m`
		`  memory: 80Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-provisioner:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-attacher:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-resizer:`
		`  requests:`
		`  cpu: 3m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-snapshotter:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  trident-autosupport:`
		`  requests:`
		`  cpu: 1m`
		`  memory: 30Mi`
		`  limits:`
		`  cpu:`
		`  memory:`

Personalizzare l'installazione dell'operatore Trident

L'operatore Trident consente di personalizzare l'installazione di Trident utilizzando gli attributi della `TridentOrchestrator` specifica. Se si desidera personalizzare l'installazione oltre gli `TridentOrchestrator` argomenti consentiti, utilizzare `tridentctl` per generare manifesti YAML personalizzati da modificare in base alle necessità.

Comprensione dei pod controller e dei pod di nodi

Trident viene eseguito come un singolo pod controller e un pod nodo su ciascun nodo worker nel cluster. Il pod del nodo deve essere in esecuzione su qualsiasi host in cui si desidera potenzialmente montare un volume Trident .

Kubernetes "selettori di nodi" e. "tollerazioni e contamini" vengono utilizzati per vincolare l'esecuzione di un pod su un nodo specifico o preferito. Utilizzo di`ControllerPlugin` e. `NodePlugin`, è possibile specificare vincoli e override.

- 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.

Opzioni di configurazione



`spec.namespace` Viene specificato in per indicare lo spazio dei nomi in `TridentOrchestrator` cui è installato Trident. Questo parametro **non può essere aggiornato dopo l'installazione di Trident**. Se si tenta di eseguire questa operazione, `TridentOrchestrator` lo stato viene modificato in `Failed`. Trident non deve essere migrato tra namespace.

Questa tabella è dettagliata `TridentOrchestrator` attributi.

Parametro	Descrizione	Predefinito
`namespace`	Spazio dei nomi in cui installare Trident	`"default"`
`debug`	Attivare il debug per Trident	`false`
`enableForceDetach`	`ontap-san`, `ontap-san-economy` , `ontap-nas` , E `ontap-nas-economy` soltanto. Funziona con Kubernetes Non-Graceful Node Shutdown (NGNS) per consentire agli amministratori del cluster di migrare in modo sicuro i carichi di lavoro con volumi montati su nuovi nodi nel caso in cui un nodo non funzioni correttamente. Per informazioni, vedere"Automazione del failover delle applicazioni stateful con Trident" .	`false`

Parametro	Descrizione	Predefinito
`enableDataLIFRefresh`	Consente l'aggiornamento dei LIF dati NVMe per i volumi pubblicati quando i LIF vengono aggiunti, rimossi o viene assegnato loro un nuovo indirizzo. Si applica a `ontap-san` con `sanType: nvme` su ONTAP unificato e NVMe/TCP su ASA r2. Per ulteriori informazioni, vedere "Aggiorna i LIF dei dati NVMe".	`false`
`windows`	Impostazione su `true` Attiva l'installazione su nodi di lavoro Windows.	`false`
`cloudProvider`	Impostato su `"Azure"` quando si utilizzano identità gestite o un'identità cloud su un cluster AKS. Impostato su `"AWS"` quando si utilizza un'identità cloud su un cluster EKS. Impostare su `"GCP"` quando si utilizza un'identità cloud su un cluster GKE.	`""`
`cloudIdentity`	Impostare sull'identità del workload ("Azure.workload.Identity/client-id: Xxxxxxxx-xxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx") quando si utilizza l'identità cloud su un cluster AKS. Impostato sul ruolo AWS IAM ("'eks.amazonaws.com/role-arn: arn:AWS:iam::123456:role/Trident-role'") quando si utilizza l'identità cloud su un cluster EKS. Impostare sull'identità cloud ("'iam.gke.io/gcp-service-account: xxxx@mygcpproject.iam.gserviceaccount.com'") quando si utilizza l'identità cloud su un cluster GKE.	`""`
`gcpWIFAudience`	Audience per il token dell'account di servizio proiettato utilizzato con Google Cloud Workload Identity Federation su OpenShift. Usa il nome canonico della risorsa del provider. Per ulteriori informazioni, consulta "Configura Google Cloud NetApp Volumes con Workload Identity Federation".	`""`
`IPv6`	Installare Trident su IPv6	falso
`k8sTimeout`	Timeout per le operazioni di Kubernetes. IL `k8sTimeout` il parametro è applicabile solo per l'installazione Trident .	`180sec`
`silenceAutosupport`	Non inviare pacchetti AutoSupport a NetApp in maniera automatica	`false`
`autosupportImage`	L'immagine del contenitore per la telemetria AutoSupport	`"netapp/trident-autosupport:26.10"`
`autosupportProxy`	Indirizzo/porta di un proxy per l'invio della AutoSupport Telemetria	`"http://proxy.example.com:8888"`
`uninstall`	Un flag utilizzato per disinstallare Trident	`false`
`logFormat`	Formato di registrazione Trident da utilizzare [text,json]	`"text"`

Parametro	Descrizione	Predefinito
`tridentImage`	Immagine Trident da installare	`"netapp/trident:26.10.1"`
`imageRegistry`	Percorso al registro interno, del formato `<registry FQDN>[:port][/subpath]`	`"registry.k8s.io"`
`kubeletDir`	Percorso della directory del kubelet sull'host	`"/var/lib/kubelet"`
`wipeout`	Un elenco di risorse da eliminare per eseguire la rimozione completa di Trident	
`imagePullSecrets`	Secrets (segreti) per estrarre immagini da un registro interno	
`imagePullPolicy`	Imposta il criterio di pull dell'immagine per l'operatore Trident. I valori validi sono: `Always` per estrarre sempre l'immagine. `IfNotPresent` per estrarre l'immagine solo se non esiste già nel nodo. `Never` per non tirare mai l'immagine.	`IfNotPresent`
`controllerPluginNodeSelector`	Selettori di nodi aggiuntivi per i pod. Segue lo stesso formato di `pod.spec.nodeSelector`.	Nessuna impostazione predefinita; opzionale
`controllerPluginTolerations`	Ignora le tolleranze Kubernetes per i pod. Segue lo stesso formato di `pod.spec.Tolerations`.	Nessuna impostazione predefinita; opzionale
`nodePluginNodeSelector`	Selettori di nodi aggiuntivi per i pod. Segue lo stesso formato di `pod.spec.nodeSelector`.	Nessuna impostazione predefinita; opzionale
`nodePluginTolerations`	Ignora le tolleranze Kubernetes per i pod. Segue lo stesso formato di `pod.spec.Tolerations`.	Nessuna impostazione predefinita; opzionale
`nodePrep`	Permette a Trident di preparare i nodi del cluster Kubernetes per gestire i volumi utilizzando il protocollo storage specificato. **Attualmente, `iscsi` è l'unico valore supportato.** A partire da OpenShift 4.19, la versione minima Trident supportata per questa funzionalità è 25.06.1.	
`k8sAPIQPS`	Limite di query al secondo (QPS) utilizzato dal controller durante la comunicazione con il server API Kubernetes. Il valore Burst viene impostato automaticamente in base al valore QPS.	`100`; facoltativo
`enableConcurrency`	Consente operazioni simultanee del controller Trident per un throughput migliore. Per ulteriori informazioni sui driver supportati, vedi il xref:./trident-install/ Controller Scalability documentation.	`enableCOSI`

Parametro	Descrizione	Predefinito
Abilita il frontend Trident Container Object Storage Interface (COSI) di Trident. Richiede `enableConcurrency` che sia impostato su `true`. Trident COSI non può essere eseguito insieme al controller COSI della community. Per ulteriori informazioni, vedi "Esegui il provisioning dello storage a oggetti con COSI".	`false`	`resources`

Parametro	Descrizione	Predefinito

Imposta i limiti delle risorse Kubernetes e le richieste per il controller Trident e i pod dei nodi. È possibile configurare CPU e memoria per ogni contenitore e sidecar per gestire l'allocazione delle risorse in Kubernetes.
 	
Per ulteriori informazioni sulla configurazione delle richieste e dei limiti delle risorse, fare riferimento a"Gestione delle risorse per pod e container" .
 	
• NON modificare i nomi di alcun contenitore o campo.
• NON modificare l'indentazione: l'indentazione YAML è fondamentale per una corretta analisi.
 	
• Per impostazione predefinita non vengono applicati i limiti: solo le richieste hanno
	<pre>resources:
	  controller:
	    trident-main:
	      requests:
	        cpu: 10m
	        memory: 80Mi
	      limits:
	          cpu:
	          memory:
	    csi-provisioner:
	      requests:
	        cpu: 2m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    csi-attacher:
	      requests:
	        cpu: 2m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    csi-resizer:
	      requests:
	        cpu: 3m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    csi-snapshotter:
	      requests:
	        cpu: 2m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    trident-autosupport:
	      requests:
	        cpu: 1m
	        memory: 30Mi
	      limits:
	        cpu:
	        memory:
	  node:</pre>
		`httpsMetrics`

Parametro	Descrizione	Predefinito
Abilita HTTPS per l'endpoint delle metriche Prometheus.	falso	`hostNetwork`

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

Configurazioni di esempio

Puoi usare gli attributi inOpzioni di configurazione quando si definisce `TridentOrchestrator` per personalizzare la tua installazione.

▼ **Configurazione personalizzata di base**

Questo esempio, creato dopo aver eseguito il `cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml` comando, rappresenta un'installazione personalizzata di base:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
```

▼ **Selettori di nodo**

Questo esempio installa Trident con selettori di nodo.

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

▼ **Nodi worker di Windows**

Questo esempio, creato dopo aver eseguito il `cat deploy/crds/tridentorchestrator_cr.yaml`

comando, installa Trident su un nodo worker Windows.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  windows: true
```

▼ **Identità gestite su un cluster AKS**

In questo esempio viene installato Trident per abilitare le identità gestite su un cluster AKS.

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

▼ **Identità cloud su un cluster AKS**

In questo esempio viene installato Trident per l'utilizzo con un'identità cloud su un cluster AKS.

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

▼ **Identità cloud su un cluster EKS**

In questo esempio viene installato Trident per l'utilizzo con un'identità cloud su un cluster AKS.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
```

```
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/trident-role'"
```

▼ **Identità cloud per GKE**

In questo esempio viene installato Trident per l'utilizzo con un'identità cloud su un cluster GKE.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

▼ **Configurazione delle richieste e dei limiti delle risorse Kubernetes per il controller Trident e i pod dei nodi Trident Linux**

Questo esempio configura le richieste e i limiti delle risorse Kubernetes per i controller Trident e i pod dei nodi Trident Linux.



Avvertenza: i valori di richiesta e limite forniti in questo esempio sono solo a scopo dimostrativo. Adattare questi valori in base all'ambiente e ai requisiti del carico di lavoro.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  resources:
    controller:
      trident-main:
        requests:
```

```
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
      # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
          cpu: 3m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
```

```
            memory: 60Mi
          limits:
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
            cpu: 50m
            memory: 32Mi
```

▼ **Configurazione delle richieste e dei limiti delle risorse Kubernetes per il controller Trident e i pod dei nodi Trident Windows e Linux**

Questo esempio configura le richieste e i limiti delle risorse Kubernetes per il controller Trident e i pod dei nodi Trident Windows e Linux.



Avvertenza: i valori di richiesta e limite forniti in questo esempio sono solo a scopo dimostrativo. Adattare questi valori in base all'ambiente e ai requisiti del carico di lavoro.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  windows: true
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
        # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
```

```
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
          cpu: 3m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
            memory: 60Mi
          limits:
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
```

```
            cpu: 50m
            memory: 32Mi
      windows:
        trident-main:
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 200m
            memory: 128Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 100m
            memory: 128Mi
        liveness-probe:
          requests:
            cpu: 2m
            memory: 40Mi
          limits:
            cpu: 50m
            memory: 64Mi
```

Installare usando tridentctl

Installare Trident usando tridentctl

È possibile installare Trident utilizzando `tridentctl`. Questo processo si applica alle installazioni in cui le immagini contenitore richieste da Trident sono memorizzate o meno in un registro privato. Per personalizzare la `tridentctl` distribuzione, fare riferimento a "Personalizzare l'implementazione tridentctl".

Informazioni critiche su Trident 26.10

È necessario leggere le seguenti informazioni critiche su Trident.

**▼ informazioni aggiornate su Trident **

- Kubernetes 1.37 è supportato con Trident 26.10. Aggiorna Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipath 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.

Installare Trident utilizzando `tridentctl`

Verificate "panoramica dell'installazione" per assicurarvi di aver soddisfatto i prerequisiti di installazione e di aver selezionato l'opzione di installazione corretta per il vostro ambiente.

Prima di iniziare

Prima di iniziare l'installazione, accedi all'host Linux e verifica che stia gestendo un "Cluster Kubernetes supportato" funzionante e che tu disponga dei privilegi necessari.

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

▼ **Details**

1. Verificare la versione di Kubernetes:

```
kubectl version
```

2. Verificare i privilegi di amministratore del cluster:

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

3. Verificare che sia possibile avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il sistema di storage tramite la rete pod:

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

Fase 1: Scaricare il pacchetto di installazione di Trident

Il pacchetto di installazione di Trident crea un pod Trident, configura gli oggetti CRD utilizzati per mantenere il proprio stato e inizializza i sidecar CSI per eseguire azioni come il provisioning e il collegamento dei volumi agli host del cluster. Scaricare ed estrarre la versione più recente del programma di installazione di Trident da "La sezione *Assets* su GitHub". Aggiornare <*Trident-installer-XX.XX.X.tar.gz*> nell'esempio con la versione Trident selezionata.

```
wget https://github.com/NetApp/trident/releases/download/v26.10.0/trident-
installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.tar.gz
cd trident-installer
```

Fase 2: Installare Trident

Installare Trident nello spazio dei nomi desiderato eseguendo il `tridentctl install` comando. È possibile aggiungere ulteriori argomenti per specificare la posizione del Registro di sistema dell'immagine.

Modalità standard

```
./tridentctl install -n trident
```

Immagini in un registro

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.10 --trident
-image <your-registry>/trident:26.10.1
```

Immagini in diversi registri

```
./tridentctl install -n trident --image-registry <csi-registry>
--autosupport-image <autosupport-registry>/trident-autosupport:26.10
--trident-image <trident-registry>/trident:26.10.1
```

Lo stato dell'installazione dovrebbe essere simile a questo.

```
....
INFO Starting Trident installation.                namespace=trident
INFO Created service account.
INFO Created cluster role.
INFO Created cluster role binding.
INFO Added finalizers to custom resource definitions.
INFO Created Trident service.
INFO Created Trident secret.
INFO Created Trident deployment.
INFO Created Trident daemonset.
INFO Waiting for Trident pod to start.
INFO Trident pod started.                          namespace=trident
pod=trident-controller-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=26.10.1
INFO Trident installation succeeded.
....
```

Verificare l'installazione

È possibile verificare l'installazione utilizzando lo stato di creazione del pod o. `tridentctl`.

Utilizzo dello stato di creazione del pod

È possibile verificare se l'installazione di Trident è stata completata controllando lo stato dei pod creati:

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-679648bd45-cv2mx   6/6     Running   0          5m29s
trident-node-linux-vgc8n              2/2     Running   0          5m29s
```

Se il programma di installazione non si completa correttamente o `trident-controller-<generated id>` non ha lo stato **Running**, la piattaforma non è stata installata. Usa `-d` per "attivare la modalità di debug" e risolvere il problema.

Utilizzo di `tridentctl`

È possibile utilizzare `tridentctl` per controllare la versione di Trident installata.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.10.1        | 26.10.1        |
+----------------+----------------+
```

Configurazioni di esempio

Gli esempi seguenti forniscono configurazioni di esempio per l'installazione di Trident utilizzando `tridentctl`.

▼ **Nodi Windows**

Per abilitare l'esecuzione di Trident sui nodi Windows:

```
tridentctl install --windows -n trident
```

▼ **Forza distacco**

Per informazioni, vedere"Automazione del failover delle applicazioni stateful con Trident" .

```
tridentctl install --enable-force-detach=true -n trident
```

▼ **Abilita le operazioni simultanee del controller Trident**

Per abilitare operazioni simultanee del controller Trident e migliorare il throughput, aggiungi l' `--enable-concurrency`opzione durante l'installazione come mostrato in questo esempio. Per ulteriori informazioni sui

driver supportati, vedi il xref:./trident-install/ Controller Scalability documentation.

```
tridentctl install --enable-concurrency -n trident
```

▼ **Aggiorna i LIF dei dati NVMe**

Per aggiornare i LIF dati NVMe per i volumi pubblicati quando la configurazione del LIF cambia, installa Trident con `--enable-datalif-refresh`. Per i driver supportati e i dettagli di configurazione, consulta "Aggiorna i LIF dei dati NVMe".

```
tridentctl install --enable-datalif-refresh -n trident
```

Personalizza l'installazione di Trident tridentctl

È possibile utilizzare il programma di installazione di Trident per personalizzare l'installazione.

Informazioni sul programma di installazione

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

> Quando si installa Trident in un repository privato, se si utilizza `--image-registry` lo switch per specificare la posizione del repository, non utilizzare `/netapp/` nel percorso del repository. Ad esempio: `./tridentctl install --image-registry <image-registry> -n <namespace>`

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

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

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

- `trident-resourcequota.yaml*`

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

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

Installare utilizzando un operatore certificato OpenShift

Installare Trident utilizzando OpenShift OperatorHub

Se usate Red Hat OpenShift, potete installare NetApp Trident usando l'operatore certificato Red Hat. Utilizzate questa procedura per installare Trident dalla piattaforma contenitore di Red Hat OpenShift.

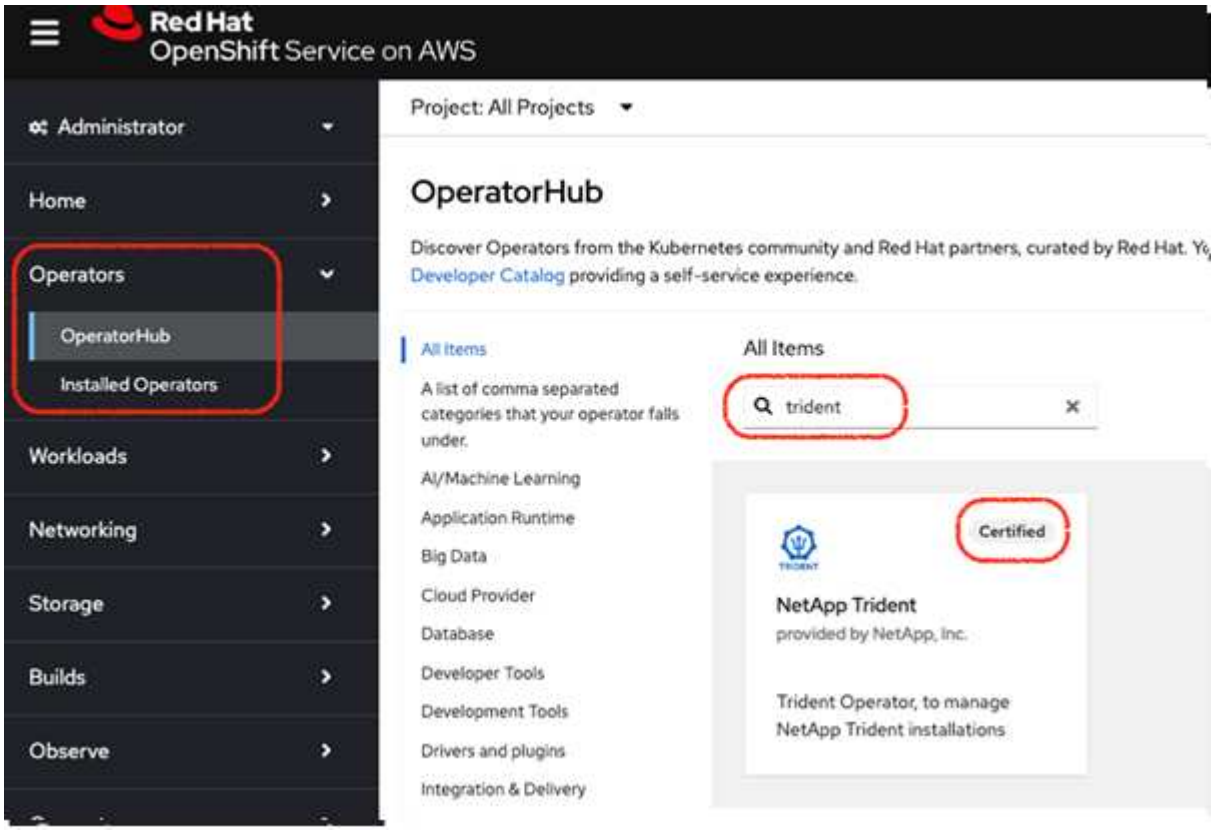
Prima di iniziare

Prima di iniziare l'installazione, "Preparare l'ambiente per l'installazione di Trident".

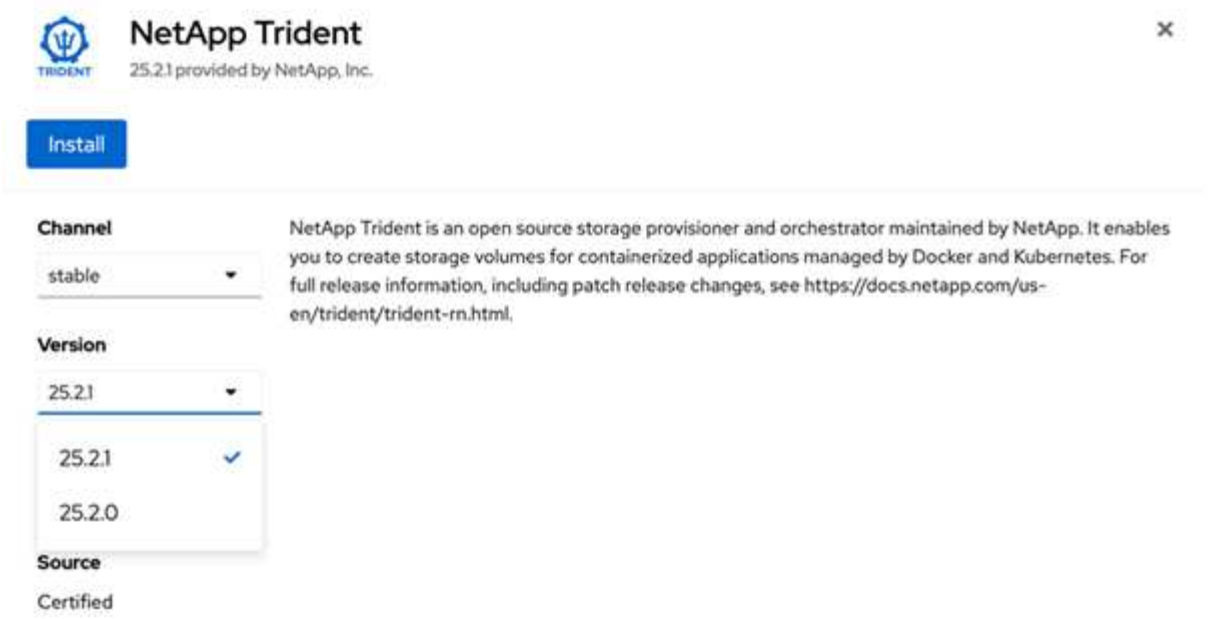
Individuare e installare l'operatore Trident

Fasi

1. Accedere a OpenShift OperatorHub e cercare NetApp Trident.

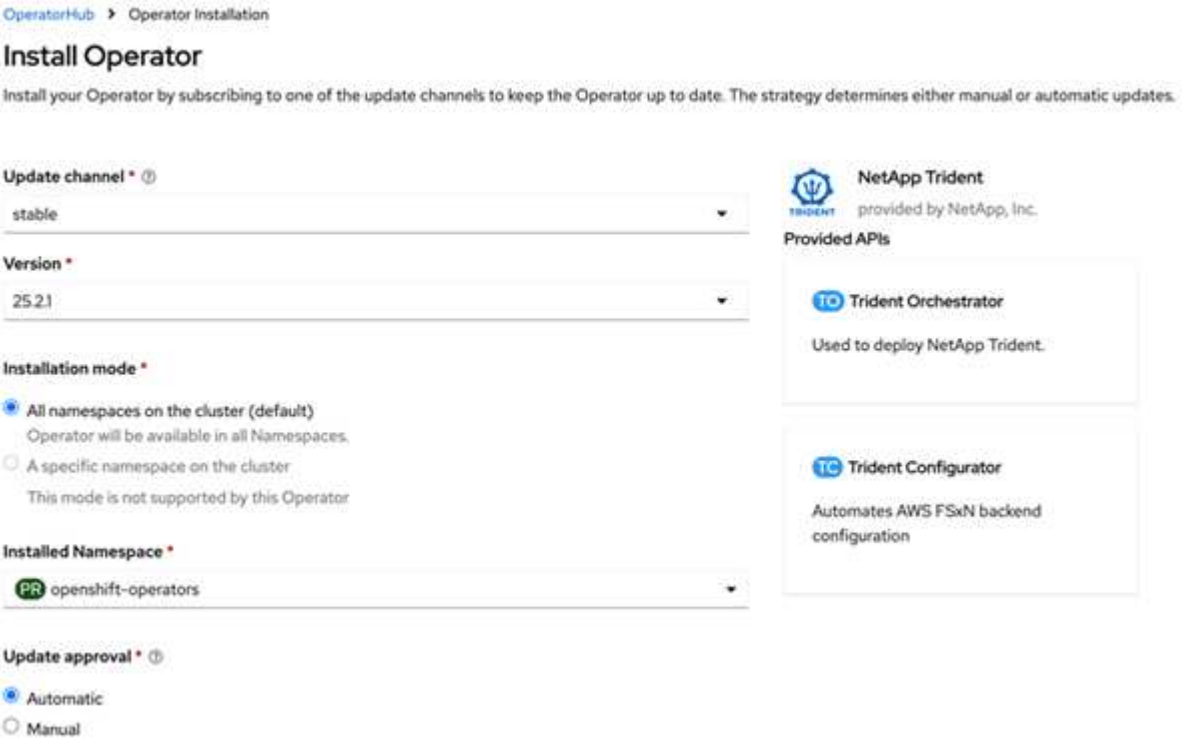


2. Fare clic su * NetApp Trident* per aprire le impostazioni di installazione.
3. Selezionare le opzioni richieste e fare clic su **Installa** per aprire la configurazione dell'operatore.

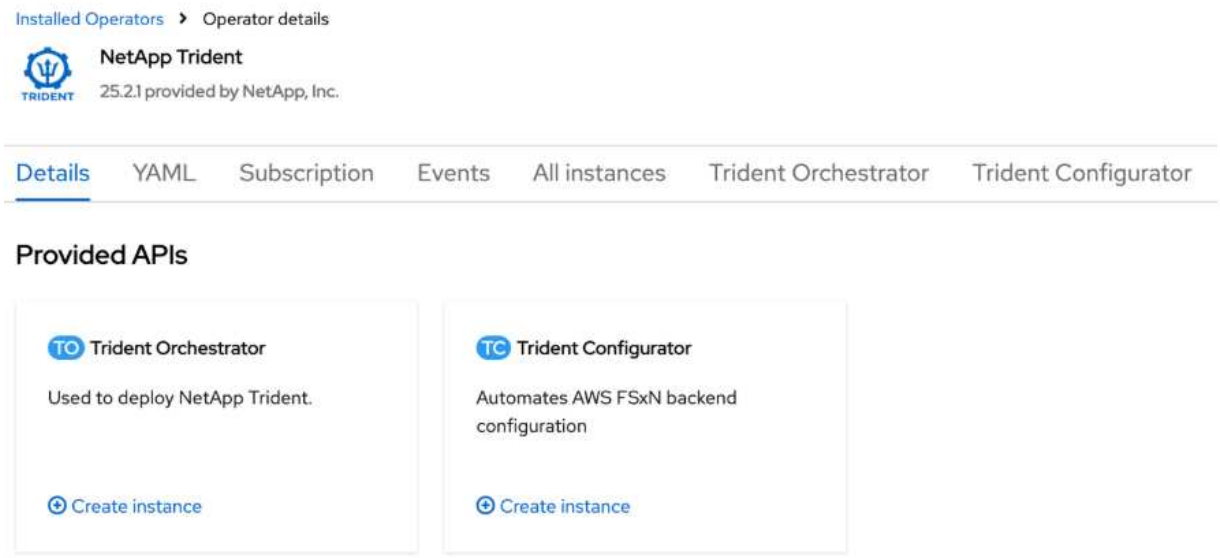


Assicurati di selezionare la versione più recente dell'operatore.

4. Mantieni tutti i parametri così come sono e fai clic su **Installa**.



5. Fare clic su **Visualizza operatore** per visualizzare i dettagli dell'operatore.



6. Fare clic su **Visualizzazione YAML** e incollare quanto segue nel modulo:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
  namespace: openshift-operators
spec:
  IPv6: false
  debug: false
  nodePrep:
  - iscsi
  imageRegistry: ''
  k8sTimeout: 180s
  namespace: trident
  silenceAutosupport: false
```

L'interfaccia utente fornisce un esempio predefinito. È possibile modificarlo direttamente anziché copiare una configurazione completa.

Opzionale: Abilita la concorrenza

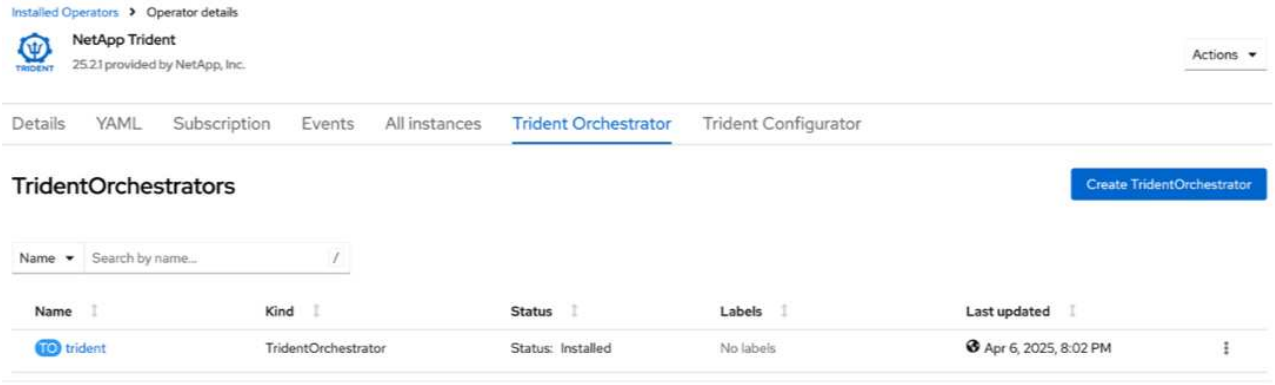
Per abilitare la concorrenza, aggiungere il seguente campo alla specifica:

```
enableConcurrency: true
```

- Red Hat Enterprise Linux CoreOS (RHCOS) non ha iSCSI abilitato e configurato.
- Puoi aggiungere il `nodePrep` parametro per configurare e abilitare i servizi iSCSI e Multipath su tutti i nodi worker OpenShift.

- A partire da OpenShift 4.19, la versione minima Trident supportata per questa funzionalità è 25.06.1.

1. Fare clic su **Crea**; Trident Orchestrator verrà installato completamente.



Disinstallare l'operatore Trident

Fasi

1. Selezionare l'operatore Trident dall'elenco degli operatori installati.
2. Selezionare se si desidera eliminare tutte le istanze dell'operando dall'operatore.

> Se non si seleziona la casella di controllo **Elimina tutte le istanze dell'operando da questo operatore**, Trident non verrà disinstallato.

3. Fare clic su **Disinstalla**.

Passa all'operatore Trident certificato OpenShift

È possibile passare all'operatore Trident certificato Red Hat OpenShift dall'operatore della community, da un'installazione basata su Helm o da un operatore distribuito manualmente. La procedura per ciascun metodo prevede la disinstallazione dell'operatore esistente e la successiva installazione dell'operatore certificato tramite OperatorHub.

Prima di iniziare

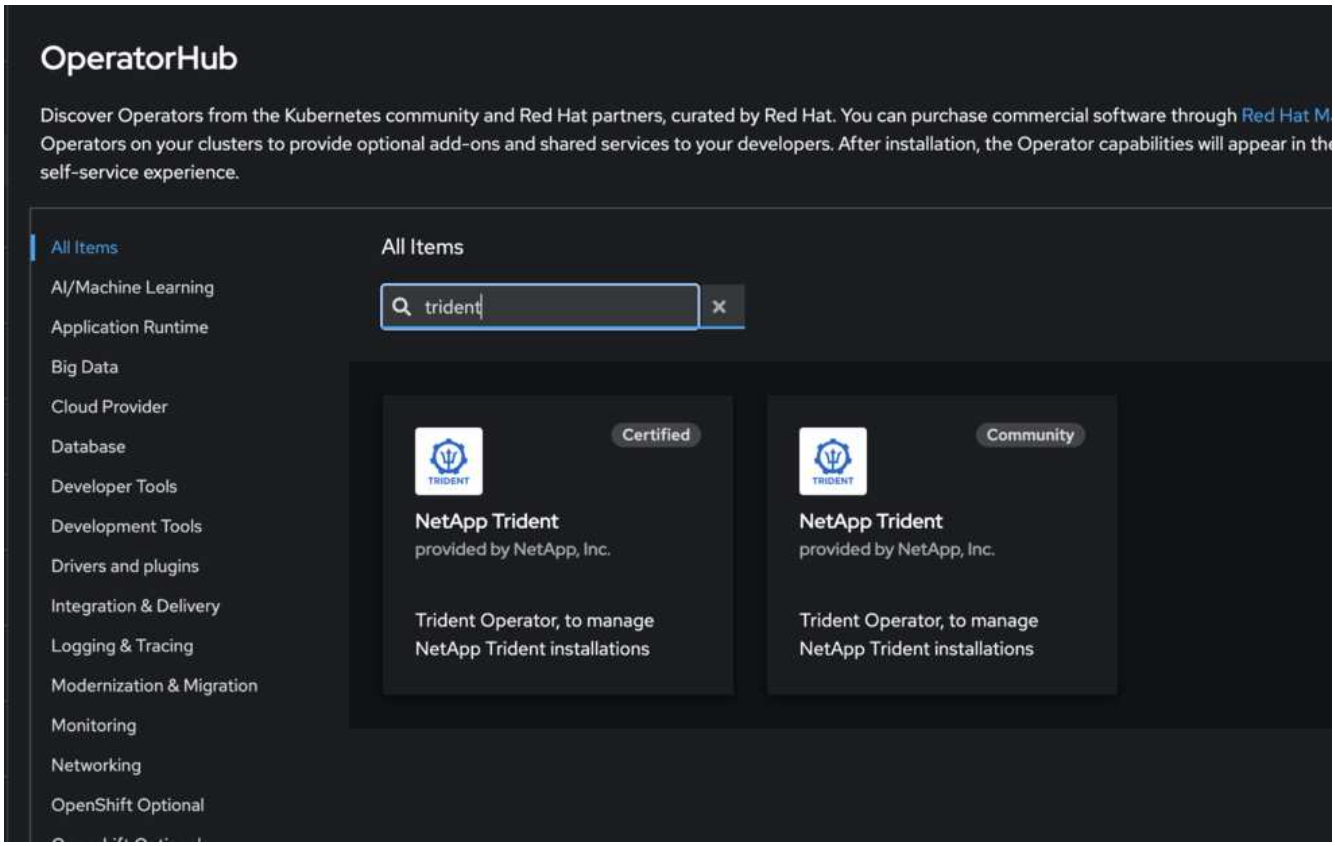
Prima di iniziare l'installazione, "Preparare l'ambiente per l'installazione di Trident".

> Non eliminare la `TridentOrchestrator` risorsa personalizzata (CR) durante il processo di disinstallazione. La `TridentOrchestrator` CR conserva la configurazione del backend e della classe di archiviazione, necessaria dopo l'installazione dell'operatore certificato.

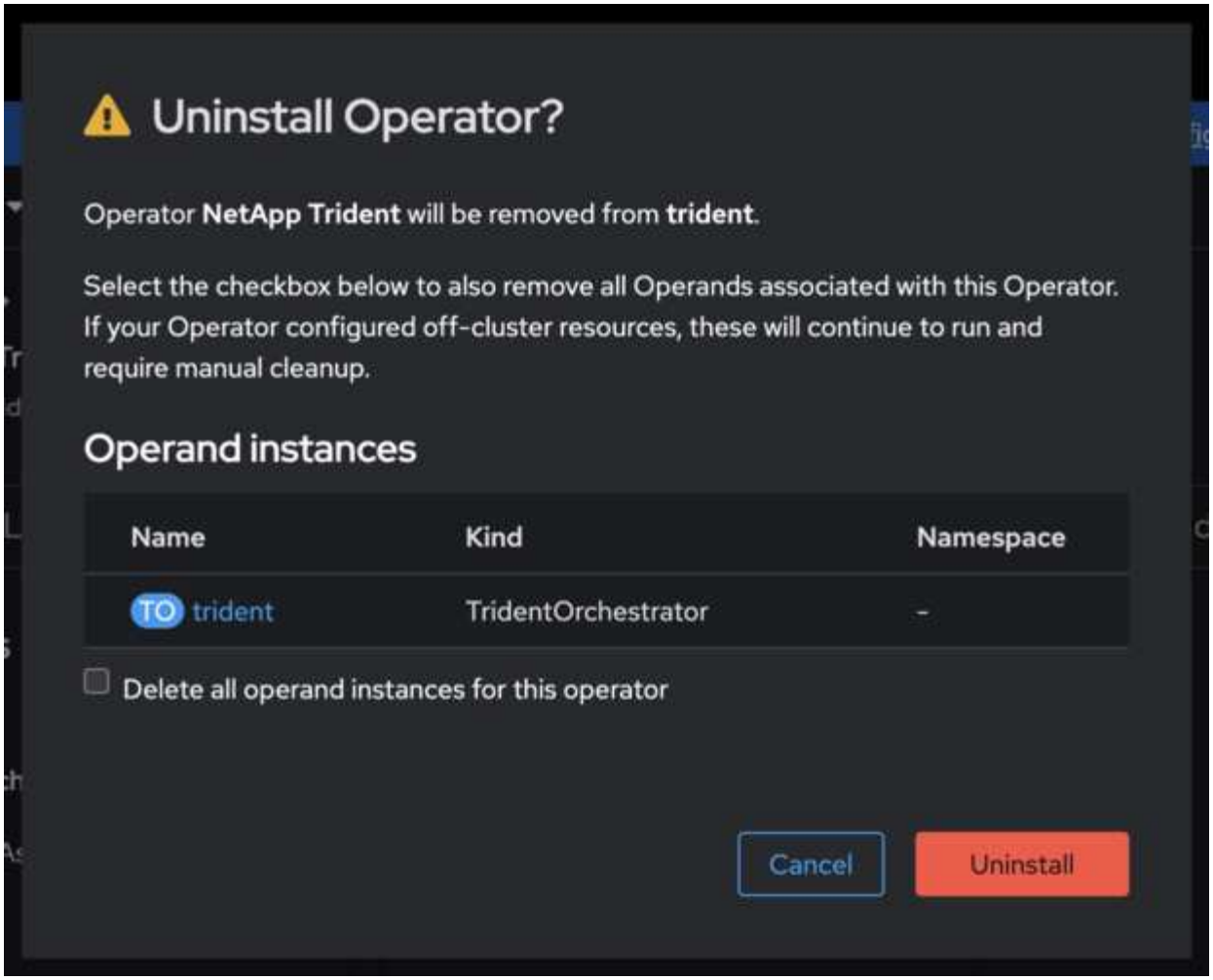
Passa dall'operatore della community

Fasi

1. Utilizzare la console OpenShift per passare all'OperatorHub.



2. Trova l’operatore della community NetApp Trident.



Non selezionare **Elimina tutte le istanze dell'operando da questo operatore**.

3. Fare clic su **Disinstalla**.
4. Dopo il completamento della disinstallazione, procedere a Installa l'operatore certificato OpenShift.

Passa da un'installazione dell'operatore basata su Helm

Fasi

1. Elenca la release di Helm per la tua installazione di Trident:

```
helm ls -n trident
```

2. Disinstalla la release Helm:

```
helm uninstall <release-name> -n trident
```

3. Dopo il completamento della disinstallazione, procedere a Installa l'operatore certificato OpenShift.

Passa da un operatore distribuito manualmente

Se hai installato Trident distribuendo manualmente l'operatore utilizzando un `bundle.yaml` dal pacchetto di installazione, rimuovilo eliminando lo stesso manifesto.

Fasi

1. Elimina la distribuzione dell'operatore utilizzando il manifesto del bundle:

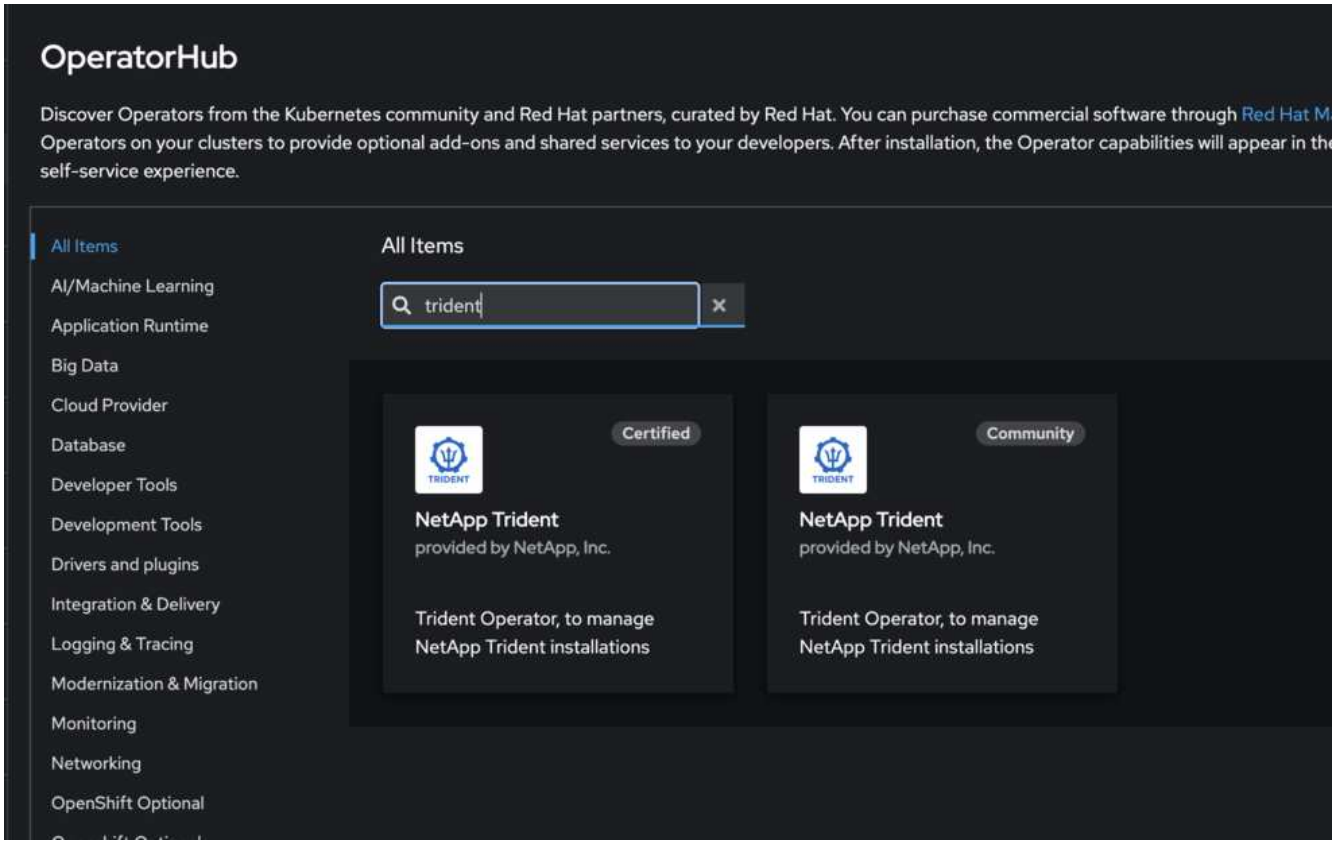
```
kubectl delete -f deploy/bundle.yaml -n trident
```

2. Dopo il completamento della disinstallazione, procedere a Installa l'operatore certificato OpenShift.

Installa l'operatore certificato OpenShift

Fasi

1. Accedere a Red Hat OperatorHub.
2. Cercare e selezionare l'operatore NetApp Trident.



3. Seguire le istruzioni visualizzate sullo schermo per installare l'operatore.

Verifica

- Controllare l'OperatorHub nella console per assicurarsi che il nuovo operatore certificato sia stato installato correttamente.

USA Trident

Prepara il nodo worker di Trident

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, NVMe/TCP o FC 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 Red Hat Enterprise Linux CoreOS (RHCOS) hanno gli strumenti installati per impostazione predefinita.

Strumenti NFS

"Installare gli strumenti NFS"se stai utilizzando: `ontap-nas` , `ontap-nas-economy` , `ontap-nas-flexgroup` , O `azure-netapp-files` .

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).

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

Strumenti SCSI su FC

Fare riferimento a "Modalità di configurazione degli host SAN FC FC-NVMe" per ulteriori informazioni sulla configurazione degli host SAN FC e FC-NVMe.

"Installare gli strumenti FC" Se si utilizza `ontap-san` con sanType `fcp` (SCSI su FC).

Punti da considerare: * SCSI su FC è supportato negli ambienti OpenShift e KubeVirt. * SCSI su FC non è supportato su Docker. * La riparazione automatica iSCSI non è applicabile a SCSI su FC.

Strumenti SMB

"Preparatevi al provisioning dei volumi SMB" se stai utilizzando: `ontap-nas` per fornire volumi SMB.

Rilevamento del servizio del nodo

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

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

Trident identifica i servizi abilitati per ogni 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

Trident è in grado di stabilire automaticamente una sessione iSCSI, eseguire la scansione dei LUN e rilevare dispositivi multipath, formattarli e montarli su un pod.

Funzionalità di riparazione automatica di iSCSI

Per i sistemi ONTAP, Trident esegue la riparazione automatica 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. Trident determina le priorità di riparazione e quando prevenire le riparazioni.
3. **Eseguire le riparazioni** necessarie per riportare lo stato della sessione iSCSI corrente allo stato della sessione iSCSI desiderato.



I log dell'attività di autoriparazione si trovano nel `trident-main` contenitore sul rispettivo pod Daemonset. Per visualizzare i log, è necessario impostare `debug` su "true" durante l'installazione di Trident.

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

- Sessioni iSCSI obsolete o non funzionanti che potrebbero verificarsi dopo un problema di connettività di rete. Nel caso di una sessione obsoleta, 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

Punti da considerare prima di aggiornare Trident

- Se sono in uso solo gli igroup per nodo (introdotti in 23,04+), la riparazione automatica iSCSI avvierà la riccansione SCSI per tutti i dispositivi nel bus SCSI.
- Se sono in uso solo gli igroup con ambito backend (deprecati da 23,04), la riparazione automatica iSCSI avvierà la riccansione SCSI per gli ID LUN esatti nel bus SCSI.
- Se si utilizza una combinazione di igroup per nodo e igroup con ambito backend, la riparazione automatica iSCSI avvierà la riccansione SCSI per gli ID LUN esatti nel bus SCSI.

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/Red Hat Enterprise Linux CoreOS (RHCOS) con iSCSI PVS, specificare il `discard` mount Option in StorageClass per eseguire il recupero dello spazio in linea. Fare riferimento alla "Documentazione di Red Hat".
- Assicurati di aver aggiornato all'ultima versione di `multipath-tools` .

RHEL 8+

1. Installare i seguenti pacchetti di sistema:

```
sudo yum install -y lsscsi iscsi-initiator-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 che `/etc/multipath.conf` contenga `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 che `/etc/multipath.conf` contenga `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

È possibile configurare le seguenti impostazioni di riparazione automatica iSCSI 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-100.2610.0.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
```

Trident modifica il `ctrl_device_tmo` valore per garantire che NVMe non ceda sul percorso in caso di arresti. Non modificare questa impostazione.

SCSI su volumi FC

Da oggi è possibile utilizzare il protocollo Fibre Channel (FC) con Trident per il provisioning e la gestione delle risorse di storage sul sistema ONTAP.

Prerequisiti

Configurare le impostazioni di rete e del nodo richieste per FC.

Impostazioni di rete

1. Ottenere il WWPN delle interfacce di destinazione. Per ulteriori informazioni, fare riferimento "visualizzazione dell'interfaccia di rete" a.
2. Ottenere il WWPN per le interfacce su iniziatore (host).

   Fare riferimento alle utility del sistema operativo host corrispondenti.
3. Configurare lo zoning sullo switch FC utilizzando i WWPN dell'host e della destinazione.

   Per informazioni, fare riferimento alla documentazione relativa del fornitore dell'interruttore.

   Per ulteriori informazioni, consultare la seguente documentazione di ONTAP:

   - "Panoramica dello zoning FCoE e Fibre Channel"
   - "Modalità di configurazione degli host SAN FC FC-NVMe"

Installare gli strumenti FC

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

- Quando si utilizzano nodi di lavoro che eseguono RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) con FC PVS, specificare il `discard` mount Option in StorageClass per eseguire il recupero dello spazio in linea. Fare riferimento alla "Documentazione di Red Hat".

RHEL 8+

1. Installare i seguenti pacchetti di sistema:

   ```
   sudo yum install -y lsscsi device-mapper-multipath
   ```

2. Abilitare il multipathing:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Assicurarsi che `/etc/multipath.conf` contenga `find_multipaths no` sotto `defaults`.

3. Assicurarsi che `multipathd` sia in esecuzione:

```
sudo systemctl enable --now multipathd
```

Ubuntu

1. Installare i seguenti pacchetti di sistema:

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. 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 che `/etc/multipath.conf` contenga `find_multipaths no` sotto `defaults`.

3. Assicurarsi che `multipath-tools` sia attivato e in esecuzione:

```
sudo systemctl status multipath-tools
```

Preparatevi al provisioning dei volumi SMB

È possibile eseguire il provisioning dei volumi SMB utilizzando `ontap-nas` conducenti.

> Devi configurare i protocolli NFS e SMB/CIFS nella SVM per creare un `ontap-nas-economy` volume SMB per i cluster on-premise ONTAP. La mancata configurazione di uno di questi

protocolli causerà un errore nella creazione del volume SMB.

`autoExportPolicy` Non è supportato per i volumi 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 2022. Trident supporta volumi SMB montati su pod in esecuzione solo sui nodi Windows.
- Almeno un segreto Trident contenente le credenziali di Active Directory. Per generare 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, puoi facoltativamente creare una condivisione SMB oppure Trident può crearne una.

   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 uno dei seguenti elementi: Il nome di una condivisione SMB creata utilizzando la console di gestione Microsoft o l'interfaccia CLI di ONTAP; un nome che consente a 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.**	""

Esegui il provisioning dello storage a oggetti con COSI

Utilizza l'interfaccia COSI (Container Object Storage Interface) per effettuare il provisioning dei bucket S3 per i carichi di lavoro Kubernetes tramite le risorse Kubernetes gestite da Trident.

Panoramica

COSI collega i provider di storage a oggetti ai carichi di lavoro Kubernetes. Trident offre le seguenti risorse:

- `TridentObjectStore`: Collega Trident a un sistema di storage a oggetti e alle relative credenziali. Crea questa risorsa nel namespace Trident.
- `BucketClass`: Definisce la selezione dell'archivio di oggetti e la policy di eliminazione dei bucket.
- `BucketClaim`: Richiede un bucket.
- `BucketAccessClass`: Definisce come viene eseguito il provisioning dell'accesso ai bucket.
- `BucketAccess`: Richiede le credenziali per uno o più bucket.

Trident supporta COSI con i seguenti driver di storage a oggetti:

- StorageGRID: generalmente disponibile.
- ONTAP S3: Anteprima tecnica.

Trident supporta il protocollo S3. Nella release iniziale, Trident supporta solo l'indirizzamento S3 in stile path.

Requisiti e limitazioni

- Abilita entrambi `enableConcurrency` e `enableCOSI` nella `TridentOrchestrator` risorsa personalizzata (TORC). COSI non è abilitato per impostazione predefinita.
- Prima di abilitare la concorrenza del controller, verifica che tutti i backend Trident configurati usino driver supportati. Un driver backend non supportato impedisce l'avvio di Trident; vedi "Prima di abilitare la scalabilità del controller".
- Non eseguire Trident COSI insieme al controller COSI della community.
- La versione iniziale utilizza l' `objectstorage.k8s.io/v1alpha2`API COSI.
- Le risorse di StorageGRID `TridentObjectStore` utilizzano credenziali root del tenant a lunga durata. Ciascuna `TridentObjectStore` si connette a un tenant StorageGRID; più risorse possono connettersi allo stesso tenant. Archivia le credenziali in un Secret di Kubernetes nel namespace Trident e proteggile in base ai tuoi requisiti di sicurezza.
- Un StorageGRID `BucketAccess` supporta fino a otto bucket.

Per informazioni generali sulle operazioni simultanee del controller, vedi "Scalabilità del controller".

Abilita COSI

Imposta entrambe le opzioni nella `TridentOrchestrator` spec:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableConcurrency: true
  enableCOSI: true
```

Applica il TORC o aggiorna la risorsa esistente utilizzando il normale flusso di lavoro dell'operatore. Per ulteriori informazioni sulla configurazione del TORC, consulta "Personalizzare l'installazione dell'operatore Trident".

Configurare un archivio di oggetti StorageGRID

Crea un Secret di Kubernetes con le credenziali del tenant StorageGRID nel namespace Trident:

```
apiVersion: v1
kind: Secret
metadata:
  name: tenant3
  namespace: trident
type: Opaque
stringData:
  username: <tenant-root-username>
```

```
  password: <tenant-root-password>
```

Crea un `TridentObjectStore` che fa riferimento al Secret. Sostituisci gli endpoint di esempio e l'ID dell'account con i valori del tuo ambiente StorageGRID:

```
apiVersion: trident.netapp.io/v1
kind: TridentObjectStore
metadata:
  name: trident-tenant3
  namespace: trident
spec:
  accountId: "<account-id>"
  credentials:
    secretName: tenant3
  labels:
    tenantInstance: three
  managementEndpoint: https://<storagegrid-management-endpoint>
  objectStoreDriver: storage-grid
  s3Endpoint: https://<storagegrid-s3-endpoint>
```

Applica entrambe le risorse:

```
kubectl apply -f tenant-secret.yaml
kubectl apply -f trident-object-store.yaml
```

Configura un archivio di oggetti ONTAP

Il supporto di ONTAP S3 tramite COSI è una Tech Preview in Trident 26.10 e non è supportato per l'uso in produzione. I campi di configurazione e il comportamento possono cambiare prima della disponibilità generale.

Crea un Secret di Kubernetes con le credenziali utente ONTAP S3 nel namespace Trident, quindi crea un `TridentObjectStore`che faccia riferimento al Secret. Sostituisci i segnaposto con i valori per il tuo ambiente ONTAP S3:

```
apiVersion: v1
kind: Secret
metadata:
  name: ontap-s3-credentials
  namespace: trident
type: Opaque
stringData:
  username: "<user>"
  password: "<password>"
```

```
---
apiVersion: trident.netapp.io/v1
kind: TridentObjectStore
metadata:
  name: ontap-s3
  namespace: trident
spec:
  objectStoreDriver: ontap-s3
  objectStoreName: ontap-object-store
  managementEndpoint: "<management-endpoint>"
  svmName: "<svm-name>"
  storagePrefix: trident
  credentials:
    secretName: ontap-s3-credentials
```

Il `storagePrefix` è facoltativo e, se omesso, il valore predefinito è `trident`. Applica il Secret e `TridentObjectStore`:

```
kubectl apply -f ontap-s3-object-store.yaml
```

Crea una BucketClass e richiedi un bucket

Un `BucketClass` può selezionare un archivio di oggetti utilizzando un selettore di etichette, un tipo di archivio di oggetti o entrambi. Senza parametri di selezione, Trident può selezionare tra tutti gli archivi di oggetti configurati.

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketClass
metadata:
  name: sg-delete
spec:
  driverName: cosi.trident.netapp.io
  deletionPolicy: Delete
  parameters:
    selector: tenantInstance=three
    objectStoreType: storage-grid
```

Crea un `BucketClaim` oggetto che utilizzi la classe:

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketClaim
metadata:
  name: claim-a1
  namespace: default
```

```
spec:
  bucketClassName: sg-delete
  protocols:
    - S3
```

Quando Trident elabora una richiesta, seleziona un archivio di oggetti corrispondente. Se più archivi corrispondono, Trident ne sceglie uno a caso per distribuire i bucket tra gli archivi corrispondenti.

Concedi a un workload l'accesso a un bucket

Crea un `BucketAccessClass` per definire il provisioning degli accessi:

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketAccessClass
metadata:
  name: sg-access
spec:
  authenticationType: Key
  driverName: cosi.trident.netapp.io
  multiBucketAccess: MultipleBuckets
```

Crea un `BucketAccess` che fa riferimento al bucket claim:

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketAccess
metadata:
  name: sg-claim-a1
  namespace: default
spec:
  bucketAccessClassName: sg-access
  bucketClaims:
    - accessMode: ReadWrite
      accessSecretName: sg-claim-a1
      bucketClaimName: claim-a1
  protocol: S3
```

Trident crea un Secret di Kubernetes contenente le informazioni di accesso al bucket. Un workload può utilizzare questo Secret tramite i meccanismi standard dei Secret di Kubernetes. Il Secret generato include i dettagli di connessione a S3, come l'identificativo del bucket, l'endpoint, la regione e le chiavi di accesso. Non esporre il Secret generato a utenti o workload non autorizzati.

Il Secret generato include chiavi come `COSI_S3_ACCESS_KEY_ID`, `COSI_S3_ACCESS_SECRET_KEY`, `COSI_S3_ADDRESSING_STYLE`, `COSI_S3_BUCKET_ID`, `COSI_S3_ENDPOINT` e `COSI_S3_REGION`.

Configurare e gestire i backend

Configurare i backend di Trident

Un backend definisce la relazione tra Trident e un sistema di storage. Spiega a Trident come comunicare con quel sistema storage e come Trident dovrebbe eseguire il provisioning dei volumi da esso.

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

- "Configurare un backend Azure NetApp Files"
- "Configurare un backend Google Cloud NetApp Volumes"
- "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"
- "USA Trident con Amazon FSX per NetApp ONTAP"

Azure NetApp Files

Configurare un backend Azure NetApp Files

Utilizza Azure NetApp Files come backend per Trident. Questo backend supporta volumi NFS e SMB. Trident supporta identità gestite e workload identity per i cluster Azure Kubernetes Service (AKS).

Ambienti cloud Azure supportati

Trident supporta backend Azure NetApp Files in più ambienti cloud di Azure.

I cloud Azure supportati includono:

- Azure Commercial
- Azure Government (Azure Government / MAG)
- Azure Cina

Quando distribuisci Trident o configuri un backend Azure NetApp Files, assicurati che Azure Resource Manager e gli endpoint di autenticazione corrispondano al tuo ambiente cloud di Azure.

Esaminare il supporto del driver Azure NetApp Files

Trident fornisce il seguente driver di storage Azure NetApp Files.

Le modalità di accesso supportate includono *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX) e *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 sulla revisione

- Azure NetApp Files non supporta volumi inferiori a 50 GiB. Trident crea un volume da 50 GiB quando viene richiesto un volume più piccolo.
- Trident supporta volumi SMB montati su pod in esecuzione solo sui nodi Windows.
- Le distribuzioni di Azure NetApp Files nei cloud Azure non commerciali richiedono endpoint di Azure Resource Manager e di autenticazione specifici per il cloud. Assicurarsi che Trident e qualsiasi configurazione di backend utilizzino gli endpoint appropriati per l'ambiente cloud Azure.

Usa identità gestite per AKS

Trident supporta "identità gestite" per i cluster AKS.

Se si utilizza `tridentctl` per creare o gestire backend di Azure NetApp Files, assicurarsi che sia configurato per il corretto ambiente cloud di Azure.

Per utilizzare le identità gestite, è necessario avere:

- Un cluster Kubernetes implementato utilizzando AKS
- Identità gestite configurate sul cluster Kubernetes AKS
- Trident installato con `cloudProvider` impostato su `"Azure"`

Operatore Trident

Modifica `tridentorchestrator_cr.yaml` e imposta `cloudProvider` su `"Azure"`.

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

Timone

Il seguente esempio installa Trident e imposta `cloudProvider` utilizzando la variabile di ambiente `$CP`:

```
helm install trident trident-operator-100.2610.0.tgz --create-namespace
--namespace <trident-namespace> --set cloudProvider=$CP
```

<code>tridentctl</code>

Il seguente esempio installa Trident e imposta il flag `cloud-provider` su `Azure`:

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

Usa l'identità del carico di lavoro per AKS

Workload identity consente ai pod Kubernetes di accedere alle risorse di Azure autenticandosi come workload identity.

Se si utilizza `tridentctl` per creare o gestire backend di Azure NetApp Files, assicurarsi che sia configurato per il corretto ambiente cloud di Azure.

Per utilizzare l'identità del carico di lavoro, è necessario disporre di:

- Un cluster Kubernetes implementato utilizzando AKS
- Identità del workload e issuer oidc configurati nel cluster AKS Kubernetes
- Trident installato con `cloudProvider` impostato su `"Azure"` e `cloudIdentity` impostato sul valore dell'identità del carico di lavoro

Operatore Trident

Modifica `tridentorchestrator_cr.yaml` e imposta `cloudProvider` su `"Azure"`. Imposta `cloudIdentity` su `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

```
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' # Edit
```

Timone

Imposta 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'"
```

Il seguente esempio installa Trident e imposta `cloudProvider` usando `$CP` e imposta `cloudIdentity` usando `$CI`:

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

<code>tridentctl</code>

Imposta 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"
```

Il seguente esempio installa Trident e imposta `cloud-provider` a `$CP` e `cloud-identity` a `$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.

Ambienti cloud Azure supportati

Trident supporta backend Azure NetApp Files in più ambienti cloud di Azure.

I cloud Azure supportati includono:

- Azure Commercial
- Azure Government (Azure Government / MAG)
- Azure Cina

Durante la preparazione dell'ambiente, assicurati che la sottoscrizione di Azure, la configurazione dell'identità e le risorse di Azure NetApp Files vengano create nell'ambiente cloud di Azure appropriato.

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 impostare 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" backend, è necessario quanto segue:

- `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.

- Le distribuzioni di Azure NetApp Files nei cloud Azure non commerciali richiedono endpoint di Azure Resource Manager e di autenticazione specifici per il cloud. Assicurarsi che Trident e qualsiasi configurazione di backend utilizzino gli endpoint appropriati per l'ambiente cloud Azure.

- 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".
  - A "Ruolo di collaboratore personalizzato" al livello di sottoscrizione (`assignableScopes`) con le seguenti autorizzazioni che sono limitate solo a ciò che Trident richiede. 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/re
ad",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/wr
```

```
ite",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/de
lete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTargets
/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistratio
ns/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistratio
ns/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistratio
ns/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": []
      }
    ]
  }
}
```

- L'istanza Azure `location` che contiene almeno un "subnet delegata". Il parametro `location` è obbligatorio al livello superiore del file di configurazione del backend. 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 2022. Trident supporta volumi SMB montati su pod in esecuzione solo sui nodi Windows.
- Almeno un segreto Trident contenente le credenziali di Active Directory in modo che Azure NetApp Files possa autenticarsi ad Active Directory. Per generare 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

Trident utilizza la configurazione del backend (subnet, rete virtuale, livello di servizio e posizione) per creare volumi Azure NetApp Files su pool di capacità disponibili nella posizione richiesta e che corrispondono al livello di servizio e alla subnet richiesti.

I backend Azure NetApp Files forniscono queste opzioni di configurazione.

Parametro	Descrizione	Predefinito
`version`	Versione della configurazione backend.	Sempre 1
`storageDriverName`	Nome del driver di storage	"azure-netapp-files"
`backendName`	Nome personalizzato per il backend di 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.	

Parametro	Descrizione	Predefinito
`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)
`location`	Nome della posizione di Azure in cui verranno creati i nuovi volumi Opzionale quando le identità gestite sono abilitate su un cluster AKS.	
`cloudConfiguration.cloudName`	Identifica l'ambiente cloud di Azure per il backend. Imposta su `AzureGovernment` quando utilizzi Azure Government.	
`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`	Insieme di funzionalità VNet per un volume, può essere `Basic` o `Standard`. Network Features non è disponibile in tutte le regioni e potrebbe dover essere abilitato in un abbonamento. Specificare `networkFeatures` quando la funzionalità non è abilitata causa il fallimento del provisioning del volume.	""

Parametro	Descrizione	Predefinito
`nfsMountOptions`	Controllo dettagliato delle opzioni di montaggio NFS. Ignorato per i volumi SMB. Per montare volumi utilizzando NFS versione 4.1, includere `nfsvers=4` nell'elenco delle opzioni di montaggio separate da virgole per scegliere NFS v4.1. Le opzioni di montaggio impostate in una definizione di storage class sovrascrivono le opzioni di montaggio impostate nella configurazione del backend.	"nfsvers=3"
`limitVolumeSize`	Il provisioning non riesce se le dimensioni del volume richiesto sono superiori a questo valore	"" (non applicato per impostazione predefinita)
`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`
`supportedTopologies`	Rappresenta un elenco di aree e zone supportate da questo backend. Per ulteriori informazioni, fare riferimento a "Utilizzare la topologia CSI".	
`qosType`	Rappresenta il tipo di QoS: automatico o manuale.	Auto
`maxThroughput`	Imposta la velocità massima consentita in MiB/sec. Supportato solo per pool di capacità QoS manuali.	`4 MiB/sec`



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

Considera gli ambienti cloud di Azure (26.06)

A partire dalla versione 26.06, Trident supporta la creazione e la gestione di backend Azure NetApp Files in più ambienti cloud di Azure.

I cloud Azure supportati includono:

- Azure Commercial
- Azure Government (Azure Government / MAG)
- Azure Cina

Quando distribuisci Trident o crei un backend di Azure NetApp Files, assicurati che gli endpoint di Azure Resource Manager e di autenticazione corrispondano al tuo ambiente cloud di Azure. Se gli endpoint non corrispondono, `tridentctl` non può autenticare e la creazione del backend fallisce.

Per un backend di Azure Government, imposta `cloudConfiguration.cloudName` nella configurazione del backend:

```
cloudConfiguration:
  cloudName: AzureGovernment
```

Autorizzazioni e risorse richieste

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

I valori per `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork` e `subnet` possono essere specificati usando nomi brevi o completamente qualificati. I nomi completamente qualificati sono consigliati nella maggior parte delle situazioni perché i nomi brevi possono corrispondere a più risorse con lo stesso nome.

> Se la rete virtuale si trova in un gruppo di risorse diverso dall'account di archiviazione Azure NetApp Files (ANF), specificare il gruppo di risorse per la rete virtuale durante la configurazione dell'elenco resourceGroups per il backend.

I valori `resourceGroups`, `netappAccounts` e `capacityPools` sono filtri che limitano l'insieme delle risorse scoperte a quelle disponibili per questo storage backend e possono essere specificati in qualsiasi combinazione. I nomi completamente 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 predefinito dei volumi specificando le seguenti opzioni in una sezione speciale del file di configurazione. Fare riferimento a Configurazioni di esempio per i dettagli.

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`	Accesso a `.snapshot` directory	`true`, `false` (Impostato esplicitamente).
`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 configurazioni di base che lasciano la maggior parte dei parametri ai valori predefiniti. Questo è il modo più semplice per definire un backend.

▼ **Configurazione minima**

Questa è la configurazione minima assoluta del backend. Con questa configurazione, Trident rileva tutti i tuoi account NetApp, pool di capacità e subnet delegate ad Azure NetApp Files nella posizione configurata e posiziona i nuovi volumi su uno di questi pool e subnet in modo casuale. Poiché `nasType` è omesso, `nfs` si applica l'impostazione predefinita e il backend esegue 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.

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  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:
    - resource-group-1/netapp-account-1/ultra-pool
  resourceGroups:
    - resource-group-1
  netappAccounts:
    - resource-group-1/netapp-account-1
  virtualNetwork: resource-group-1/eastus-prod-vnet
  subnet: resource-group-1/eastus-prod-vnet/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 backend colloca i volumi nella posizione di Azure `eastus` in un `Ultra` pool di

capacità. Trident scopre automaticamente tutte le sottoreti delegate ad Azure NetApp Files in quella posizione e colloca un nuovo volume su una di esse in modo casuale.

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

▼ **Esempio di backend con pool di capacità QoS manuali**

Questa configurazione del backend posiziona i volumi in Azure `eastus` posizione con pool di capacità QoS manuali.

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
```

```
    defaults:
      maxThroughput: 3
```

▼ **Configurazione avanzata**

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

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
virtualNetwork: application-group-1/eastus-prod-vnet
subnet: application-group-1/eastus-prod-vnet/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 backend definisce più pool di storage in un unico file. Questo 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 li 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:
      - application-group-1/netapp-account-1/ultra-1
      - application-group-1/netapp-account-1/ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - application-group-1/netapp-account-1/premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - application-group-1/netapp-account-1/standard-1
      - application-group-1/netapp-account-1/standard-2
```

▼ **Configurazione delle topologie supportate**

Trident facilita il provisioning dei volumi per i carichi di lavoro in base alle regioni e alle zone di disponibilità. Il `supportedTopologies` blocco in questa configurazione di backend viene utilizzato per fornire un elenco di regioni e zone per backend. I valori di regione e zona specificati qui devono corrispondere ai valori di regione e zona delle etichette su ciascun nodo del cluster Kubernetes. Queste regioni e zone rappresentano l'elenco dei valori consentiti che possono essere forniti in una classe di storage. Per le classi di storage che contengono un sottoinsieme delle regioni e delle zone fornite in un backend, Trident crea i volumi nella regione e nella zona menzionate. Per ulteriori informazioni, consulta "Utilizzare la topologia CSI".

```
---
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
```

```
supportedTopologies:
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-1
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-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

Utilizzando `parameter.selector` puoi specificare per ogni `StorageClass` il pool virtuale che viene utilizzato per ospitare un volume. Il volume avrà gli aspetti definiti nel pool scelto.

```
---
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

Utilizzando `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 i pool che supportano i volumi SMB.
`nasType: nfs` o `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 utilizzi un cloud Azure non commerciale, assicurati che `tridentctl` sia configurato per utilizzare Azure Resource Manager e gli endpoint di autenticazione per il tuo ambiente cloud Azure. Se la creazione del backend non riesce, controlla la configurazione del backend e visualizza i log per determinarne la causa:

```
tridentctl logs
```

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

Google Cloud NetApp Volumes

Configurare Google Cloud NetApp Volumes

È possibile configurare Google Cloud NetApp Volumes come backend per Trident al fine di predisporre lo storage per i carichi di lavoro di Kubernetes.

Panoramica

Trident supporta Google Cloud NetApp Volumes sia per carichi di lavoro NAS (NFS e SMB) che a blocchi (iSCSI).

- I carichi di lavoro NAS utilizzano il `google-cloud-netapp-volumes` backend
- I carichi di lavoro a blocchi (iSCSI) utilizzano il `google-cloud-netapp-volumes-san` backend

I volumi NAS offrono storage basato su file e vi si accede tramite i protocolli NFS o SMB. Questi volumi supportano l'accesso condiviso tra più pod o nodi.

I volumi a blocchi forniscono storage a blocchi raw e sono accessibili come dispositivi iSCSI collegati ai nodi Kubernetes. Questi volumi vengono utilizzati quando le applicazioni richiedono l'accesso a livello di blocco.

Ciò si applica ai seguenti ambienti:

- Trident 26.02 e versioni successive
- Google Kubernetes Engine (GKE) o cluster Red Hat OpenShift in esecuzione su GCP
- OpenShift Workload Identity Federation (WIF) richiede Trident 26.10 o versioni successive
- Google Cloud NetApp Volumes pool di storage

Per configurare lo storage a blocchi (iSCSI), vedere "Configurare storage a blocchi (iSCSI)".

Prepararsi alla configurazione

L'identità cloud consente ai carichi di lavoro Kubernetes di accedere alle risorse di Google Cloud autenticandosi come identità del carico di lavoro anziché utilizzare credenziali statiche.

Prima di configurare l'identità del carico di lavoro, usa un account autorizzato a creare le risorse di identità richieste. Per GKE, questo richiede le autorizzazioni Google Cloud per creare e configurare un account di servizio Google Cloud. Per OpenShift, questo richiede l'autorizzazione a creare o selezionare la configurazione WIF del cluster e a configurare il Google Cloud Workload Identity Pool e il provider.

Usa l'identità del carico di lavoro GKE

Per utilizzare l'identità del carico di lavoro GKE con Google Cloud NetApp Volumes, devi avere:

- Un cluster Kubernetes distribuito utilizzando Google Kubernetes Engine (GKE)
- Identità del carico di lavoro abilitata sul cluster GKE e server dei metadati abilitato sui pool di nodi
- Un account di servizio Google Cloud con il ruolo di amministratore di Google Cloud NetApp Volumes (`roles/netapp.admin`) o un ruolo personalizzato equivalente
- Trident installato con il cloud provider impostato su `GCP` e l'annotazione dell'identità cloud configurata

Operatore Trident

Per installare Trident utilizzando l'operatore Trident, modificare `tridentorchestrator_cr.yaml`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  cloudProvider: "GCP"
  cloudIdentity: "iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com"
```

Timone

Imposta il cloud provider e l'identità cloud durante l'installazione di Trident con Helm:

```
helm install trident trident-operator-100.6.0.tgz \
  --set cloudProvider=GCP \
  --set cloudIdentity="iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com"
```

tridentctl

Installa Trident specificando il cloud provider e l'identità cloud:

```
tridentctl install \
  --cloud-provider=GCP \
  --cloud-identity="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com" \
```

```
  -n trident
```

Usa la federazione delle identità dei carichi di lavoro su OpenShift

A partire da Trident 26.10, puoi utilizzare Google Cloud Workload Identity Federation (WIF) per autenticare i driver di Google Cloud NetApp Volumes sui cluster OpenShift in esecuzione su GCP. Questo vale per OpenShift Container Platform (OCP, incluse le implementazioni IPI) autogestito e per OpenShift Dedicated (OSD), e supporta sia i driver NAS (`google-cloud-netapp-volumes`) sia i driver SAN (`google-cloud-netapp-volumes-san`).

WIF scambia un token dell'account di servizio Kubernetes proiettato con credenziali Google Cloud di breve durata. Trident utilizza queste credenziali per impersonare un account di servizio Google Cloud; i ruoli IAM di tale account di servizio autorizzano le sue chiamate API a Google Cloud NetApp Volumes. Ciò evita di memorizzare una chiave dell'account di servizio di lunga durata nella configurazione del backend.

> ⓘ Su OpenShift Dedicated, il provider OIDC può essere condiviso con altri componenti OpenShift. Se l'elenco del provider `allowedAudiences` è vuoto, Google accetta il nome canonico della risorsa del provider, con o senza il `https://` prefisso, quindi non devi modificare il provider per l'audience di Trident. Non impostare un elenco di audience esplicito solo per Trident. Se il provider ha già audience esplicite, aggiungi l'audience di Trident mantenendo tutti i valori esistenti. La sostituzione dell'elenco può impedire ad altri componenti OpenShift di scambiare token.

Prepara il cluster OpenShift

Per OSD, crea o seleziona una configurazione GCP WIF durante la creazione del cluster. Ad esempio, la OpenShift Cluster Manager CLI può crearne una:

```
ocm gcp create wif-config --name <wif-name> --project <gcp-project-id>
```

Seleziona Workload Identity Federation come tipo di autenticazione del cluster e associa la configurazione WIF al cluster. Per OCP autogestito, configura un provider OIDC di Google Cloud che si fidi dell'emittente del token dell'account di servizio del cluster. Le fasi di provisioning del cluster e di proiezione del token differiscono tra OSD e OCP autogestito; segui le linee guida Red Hat applicabili e rendi il token proiettato disponibile al controller Trident all'indirizzo `/var/run/secrets/wif-token/token`.

L'autenticazione WIF di OSD viene configurata quando crei il cluster. L'audience WIF di Trident è un'impostazione di installazione separata; non devi ricreare il cluster per abilitare WIF per Trident.

Prima di installare o configurare Trident, verifica che l'identità OpenShift utilizzata per l'installazione disponga delle autorizzazioni del cluster richieste dal metodo di installazione scelto. Dopo l'installazione, verifica che i pod del controller e dei nodi siano integri:

```
oc whoami
oc auth can-i create namespaces
oc auth can-i create csidrivers.storage.k8s.io
oc auth can-i create customresourcedefinitions.apiextensions.k8s.io
oc auth can-i create securitycontextconstraints.security.openshift.io
```

```
oc -n trident get pods
oc -n trident rollout status deploy/trident-controller --timeout=300s
oc -n trident rollout status ds/trident-node-linux --timeout=300s
```

Installa Trident con l'audience WIF

Trident proietta il token per WIF solo quando configuri l'audience durante l'installazione. Usa la stessa audience nel provider Google Cloud e nel backend `wipCredentialConfig.audience`:

```
//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>
```

Imposta `cloudProvider` su `GCP` e configura `gcpWIFAudience` utilizzando il metodo di installazione selezionato:

tridentctl

```
tridentctl install \
  --cloud-provider GCP \
  --gcp-wif-audience "<audience>" \
  --namespace trident
```

Operatore Trident

Imposta questi campi nella `TridentOrchestrator` specifica:

```
spec:
  cloudProvider: GCP
  gcpWIFAudience: "<audience>"
```

Timone

Imposta questi valori durante l'installazione del chart:

```
helm install trident netapp-trident/trident-operator \
  --set cloudProvider=GCP \
  --set-string gcpWIFAudience="<audience>"
```

Non impostare anche l'opzione GKE `cloudIdentity` / `--cloud-identity` per OpenShift WIF. Non applicare patch al Deployment del controller Trident dopo l'installazione; l'operator ripristina le modifiche manuali. Il token proiettato è disponibile per il backend in `/var/run/secrets/wif-token/token`. Il container `trident-main` non include una shell, quindi ispeziona la configurazione del volume del

Deployment del controller invece di usare `oc exec` per controllare il file:

```
oc -n trident get deploy trident-controller -o yaml
```

Configura la relazione di trust e le autorizzazioni di Google Cloud

Configura un Google Cloud Workload Identity Pool e un provider OIDC per il cluster. L'issuer del provider deve corrispondere al claim `iss` del token e la sua mappatura degli attributi deve mappare `assertion.sub` su `google.subject`. L'audience consentita del provider deve corrispondere al claim `aud` del token proiettato.

Gli endpoint pubblici `.well-known/openid-configuration` e JWKS sono richiesti solo quando usi il rilevamento OIDC. Google supporta anche il caricamento diretto del JWKS tramite `jwksJson` o `--jwk-json-path`.

Assegna al principal WIF il `roles/iam.workloadIdentityUser`ruolo sull'account di servizio Google Cloud che Trident impersonerà. Per un soggetto del token di `system:serviceaccount:trident:trident-controller`, il principal ha questa forma:

```
principal://iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/subject/system:serviceaccount:trident:trident-controller
```

Concedi l'autorizzazione all'account di servizio impersonato `roles/netapp.admin` sul progetto che contiene le risorse di Google Cloud NetApp Volumes, oppure un ruolo personalizzato equivalente con le autorizzazioni richieste. Si tratta di autorizzazioni separate: la prima consente l'impersonificazione, mentre la seconda autorizza le operazioni dell'API di storage.

Se l'audience del token e la configurazione del provider non corrispondono, Google Security Token Service può rifiutare lo scambio con un `invalid_grant` errore di audience. Verifica la configurazione del provider con:

```
gcloud iam workload-identity-pools providers describe <provider-id> \
  --project=<wif-pool-project-id> \
  --location=global \
  --workload-identity-pool=<pool-id> \
  --format="yaml(oidc.allowedAudiences,oidc.issuerUri,attributeMapping)"
```

Quando diagnostichi la configurazione del token, confronta i claim `iss`, `aud` e `sub` del token con la configurazione del provider e di IAM. Non condividere né registrare nei log il token proiettato completo.

Prepara lo storage Google Cloud NetApp Volumes

Crea o identifica un pool di storage nella regione utilizzata dal backend. I backend NAS richiedono un pool FILE; i backend SAN usano i pool Flex Unified descritti in "Configurare storage a blocchi (iSCSI)". Assicurati che la configurazione VPC e Private Service Access (PSA) del pool sia appropriata per la rete del cluster.

Ad esempio, elenca gli intervalli di peering VPC globali esistenti per una rete prima di selezionare l'intervallo

PSA per un pool:

```
gcloud compute addresses list \
  --project <gcnv-project-id> \
  --global \
  --filter="purpose=VPC_PEERING AND network:<network-name>" \
  --format="table(name,address,prefixLength,status,network,purpose)"
```

Usa la regione del pool in `location` e, quando necessario, seleziona esplicitamente il pool usando il percorso completo della risorsa in `storagePools`. Le incongruenze tra regione e posizione del pool, una rete errata o un pool non disponibile possono impedire il rilevamento o il provisioning anche se l'autenticazione riesce.

Configura il backend di Trident

Utilizza l'audience prevista del token sia nella configurazione del provider sia in `wipCredentialConfig.audience`. Il campo `projectNumber` identifica il progetto Google Cloud NetApp Volumes; l'audience usa il numero di progetto che contiene il Workload Identity Pool. Sostituisci i segnaposto in questo esempio con i valori del tuo ambiente:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas-wif
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
    serviceLevel: Premium
  wipCredentialConfig:
    type: external_account
    audience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
    subject_token_type: urn:ietf:params:oauth:token-type:jwt
    token_url: https://sts.googleapis.com/v1/token
    credential_source:
      file: /var/run/secrets/wif-token/token
    service_account_impersonation_url:
https://iamcredentials.googleapis.com/v1/projects/-
```

```
/serviceAccounts/<service-account-email>:generateAccessToken
```

Per un backend SAN, usa `google-cloud-netapp-volumes-san` come `storageDriverName` e le impostazioni del pool Flex Unified applicabili. La configurazione delle credenziali WIF è altrimenti la stessa per entrambi i driver.

Applica il backend e verifica il suo stato osservato:

```
oc apply -f <backend-file>.yaml
oc -n trident get tridentbackendconfig <backend-name> -o yaml
```

Conferma che il backend sia `Bound`attivo e che la sua ultima operazione sia riuscita prima di testare il provisioning. Ispeziona lo stato live `spec`e `status`; un'annotazione di configurazione applicata in precedenza potrebbe non riflettere le modifiche successive. Non includere `apiKey`credenziali statiche quando usi WIF.

Verifica il provisioning end-to-end

Crea una StorageClass che punti al backend, quindi un piccolo PVC di test. Ad esempio, una StorageClass NAS può selezionare un'etichetta di backend:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-wif-smoke
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  selector: "auth=wif"
allowVolumeExpansion: true
```

Aggiungi l'etichetta `auth: wif` corrispondente alla voce di storage del backend NAS, crea un PVC utilizzando `gcnv-wif-smoke`, e verifica che raggiunga `Bound`. Per un test SAN di base, utilizza `google-cloud-netapp-volumes-san`, le impostazioni del driver SAN e un pool Flex Unified appropriato. Elimina il PVC di test e StorageClass al termine.

Risolvi i problemi comuni

Sintomo	Cosa controllare
`invalid_grant` o mancata corrispondenza del pubblico	Confronta il token proiettato `aud` con gli audience consentiti del provider e `wipCredentialConfig.audience`.
`iam.serviceAccounts.getAccessToken` autorizzazione negata	Verifica che il principal WIF esatto per il controller Trident `sub` abbia `roles/iam.workloadIdentityUser` sull'account di servizio impersonato.

Sintomo	Cosa controllare
`netapp.storagePools.list` autorizzazione negata	Verifica che l'account di servizio impersonato disponga delle autorizzazioni per Google Cloud NetApp Volumes, ad esempio `roles/netapp.admin`, nel progetto corretto.
Nessun pool trovato o pool non trovato	Verifica il backend `location`, la regione e il tipo del pool, la rete, la configurazione PSA e qualsiasi percorso della risorsa del pool di storage configurato esplicitamente.
Il backend rimane non associato	Esamina i log del controller Trident e lo stato di TridentBackendConfig per individuare l'errore di autenticazione, autorizzazione o rilevamento del pool sottostante.

Per esaminare i log del controller alla ricerca di errori di riconciliazione del backend:

```
oc -n trident logs deploy/trident-controller -c trident-main --tail=400
```

Configurare l'archiviazione NAS

> Per i pool di storage UNIFIED di Google Cloud NetApp Volumes, Trident applica regole di denominazione e convalida specifiche di UNIFIED durante le operazioni sui volumi.

Durante la ricerca di un volume, Trident può valutare diverse varianti compatibili del nome del volume (ad esempio, formati con trattino e underscore) per migliorare l'affidabilità dell'importazione e del rilevamento.

Dettagli del driver

Trident fornisce il `google-cloud-netapp-volumes` driver per il provisioning dello storage NAS da Google Cloud NetApp Volumes.

Il driver supporta le seguenti modalità di accesso:

- ReadWriteOnce (RWO)
- ReadOnlyMany (ROX)
- ReadWriteMany (RWX)
- ReadWriteOncePod (RWOP)

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

Configurare un backend NAS Trident

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas
```

```
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
```

Effettuare il provisioning dei volumi NAS

I volumi NAS vengono forniti tramite il backend `google-cloud-netapp-volumes` e supportano i protocolli NFS e SMB.

StorageClass per volumi NFS

Per eseguire il provisioning dei volumi NFS, impostare `nasType` su `nfs`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "nfs"
allowVolumeExpansion: true
```

StorageClass for volumi SMB

Per effettuare il provisioning dei volumi SMB, impostare `nasType` su `smb` e fornire le credenziali.

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

```
allowVolumeExpansion: true
```

Esempio di PersistentVolumeClaim (RWX)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwx
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

Esempio di PersistentVolumeClaim (RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

I volumi NAS utilizzano `volumeMode: Filesystem`.

Configura Google Cloud NetApp Volumes per carichi di lavoro SAN

Usa il `google-cloud-netapp-volumes-san` driver per eseguire il provisioning di volumi a blocchi iSCSI dai pool di storage Flex Unified di Google Cloud NetApp Volumes.

Il driver SAN supporta solo carichi di lavoro a blocchi. Per eseguire il provisioning dei volumi NAS, usa il `google-cloud-netapp-volumes` driver.

Panoramica

Il driver effettua il provisioning dei volumi dai pool di storage Flex Unified e li presenta ai nodi Kubernetes come dispositivi a blocchi iSCSI.

Il driver supporta i seguenti ambienti:

- Trident 26.02 e versioni successive
- Google Kubernetes Engine (GKE) o cluster Red Hat OpenShift in esecuzione su GCP
- Google Cloud NetApp Volumes Flex pool di storage unificati
- Carichi di lavoro basati su iSCSI

A partire da Trident 26.10, il driver SAN supporta Google Cloud Workload Identity Federation (WIF) sui cluster OpenShift Container Platform (OCP, incluse le distribuzioni IPI) autogestiti e OpenShift Dedicated (OSD) in esecuzione su GCP. Per i prerequisiti di WIF e la configurazione del backend, vedi "Usa la federazione delle identità dei carichi di lavoro su OpenShift". Usa il nome del driver SAN e le impostazioni di storage Flex Unified in questa pagina.

Configura WIF per SAN

WIF scambia un token dell'account di servizio Kubernetes proiettato con credenziali Google Cloud di breve durata. Trident utilizza queste credenziali per impersonare un account di servizio Google Cloud quando chiama l'API di Google Cloud NetApp Volumes.

Configura un Google Cloud Workload Identity Pool e un provider OIDC per il cluster OpenShift. L'issuer del provider deve corrispondere al claim `iss` del token e la sua mappatura degli attributi deve mappare `assertion.sub` su `google.subject`.

Installa Trident con il nome della risorsa del provider come audience WIF. Questo fa sì che Trident proietti il token dell'account di servizio del controller in `/var/run/secrets/wif-token/token`.

Operatore Trident

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  cloudProvider: GCP
  gcpWIFAudience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
```

Timone

```
helm install trident <trident-operator-package> \
  --namespace trident \
  --create-namespace \
  --set cloudProvider=GCP \
  --set-string gcpWIFAudience="//iam.googleapis.com/projects/<wif-pool-
project-number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
```

tridentctl

```
tridentctl install \
  --namespace trident \
  --cloud-provider GCP \
  --gcp-wif-audience "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
```

Usa lo stesso pubblico per l'installazione di Trident e `wipCredentialConfig.audience`. Non configurare anche GKE `cloudIdentity` o `--cloud-identity`. Non applicare patch al Deployment del controller per aggiungere il token proiettato; il Trident operator può sovrascrivere questa modifica.

> Su OpenShift Dedicated, il provider OIDC può essere condiviso con altri componenti di OpenShift. Se l'elenco del provider `allowedAudiences` è vuoto, Google accetta il nome canonico della risorsa del provider, con o senza il `https://` prefisso, quindi non è necessario modificare il provider per il pubblico indicato sopra. Non impostare un elenco di destinatari esplicito solo per Trident. Se il provider ha già destinatari espliciti, aggiungi il destinatario di Trident mantenendo tutti i valori esistenti. La sostituzione dell'elenco può impedire ad altri componenti di OpenShift di scambiare token.

Assegna all'entità WIF per l'account di servizio del controller Trident il `roles/iam.workloadIdentityUser` ruolo nell'account di servizio Google Cloud che Trident impersonerà. Per l'account di servizio del controller Trident predefinito, il soggetto del token è `system:serviceaccount:trident:trident-controller`; utilizza il soggetto configurato per la tua installazione. Assegna all'account di servizio impersonato `roles/netapp.admin`, o autorizzazioni equivalenti, nel progetto che contiene le risorse Google Cloud NetApp Volumes.

Aggiungi la seguente sezione alla `spec` del backend SAN:

```
wipCredentialConfig:
  type: external_account
  audience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
  subject_token_type: urn:ietf:params:oauth:token-type:jwt
  token_url: https://sts.googleapis.com/v1/token
  credential_source:
    file: /var/run/secrets/wif-token/token
  service_account_impersonation_url:
https://iamcredentials.googleapis.com/v1/projects/-
/serviceAccounts/<service-account-email>:generateAccessToken
```

La configurazione WIF è la stessa per entrambi i driver GCNV. Il backend SAN deve comunque utilizzare `google-cloud-netapp-volumes-san` e un pool di storage Flex Unified supportato.

Pool di storage unificati Flex

I pool di storage Flex Unified forniscono storage a blocchi utilizzando il protocollo iSCSI e sono necessari per il provisioning SAN:

- Sono supportati i pool regionali Flex Unified.
- I pool Flex Unified ZONAL sono supportati a partire da Trident 26.02.1.
- Per i carichi di lavoro SAN è supportato esclusivamente il livello di servizio **Flex**.

Il backend `location` deve corrispondere alla regione del pool Flex Unified. Se Trident non trova il pool, verifica la regione, la rete e il percorso delle risorse del pool.

Configura un backend SAN Trident

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-san
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
      performance: flex
    network: "<vpc-network>"
    serviceLevel: Flex
```

Crea un StorageClass

Dopo aver configurato il backend SAN, crea una StorageClass che fa riferimento al driver `google-cloud-netapp-volumes-san`.

Specifica il tipo di filesystem in StorageClass, non nel backend.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes-san"
  fsType: "ext4"
```

```
allowVolumeExpansion: true
```

Il driver supporta i seguenti tipi di filesystem:

- `ext4` (predefinito)
- `ext3`
- `xfs`

> Il driver SAN supporta solo il livello di servizio Flex e non utilizza parametri backend specifici del NAS come `exportRule`, `unixPermissions`, `nasType`, `snapshotDir`, `nfsMountOptions` o impostazioni relative al tiering.

Verifica il provisioning WIF

Applica il backend e verifica che TridentBackendConfig sia associato e che la sua ultima operazione sia andata a buon fine:

```
oc apply -f <backend-file>.yaml
oc -n trident get tridentbackendconfig <backend-name> -o yaml
```

Crea un PVC utilizzando lo `gcnv-san` StorageClass di questa pagina e verifica che raggiunga `Bound`. Questo convalida contemporaneamente lo scambio WIF, l'impersonificazione dell'account di servizio, le autorizzazioni di Google Cloud NetApp Volumes e la selezione del pool SAN. Elimina il PVC di test al termine.

Se la riconciliazione del backend non riesce, esamina i log del controller Trident:

```
oc -n trident logs deploy/trident-controller -c trident-main --tail=400
```

Verifica la presenza dei seguenti errori comuni:

- Per un `invalid_grant` errore di audience, confronta la claim del token `aud` con le audience consentite del provider OIDC e `wipCredentialConfig.audience`.
- Per un `iam.serviceAccounts.getAccessToken` errore di autorizzazione, verifica che il principal WIF del controller Trident abbia `roles/iam.workloadIdentityUser` sull'account di servizio impersonato.
- Per un `netapp.storagePools.list` errore di autorizzazione, verifica che l'account di servizio impersonato disponga delle autorizzazioni necessarie per Google Cloud NetApp Volumes nel progetto corretto.
- Se non trovi il pool, verifica che la sua posizione e il suo tipo siano supportati, che la regione del backend corrisponda a quella del pool e che il backend faccia riferimento alla rete e al pool corretti.

Effettuare il provisioning dei volumi a blocchi

ReadWriteOnce (RWO)

```
apiVersion: v1
```

```
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadWriteOncePod (RWOP)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwop
spec:
  accessModes:
    - ReadWriteOncePod
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadOnlyMany (ROX)

Un modello comune per ROX è clonare un volume ReadWriteOnce esistente e montare il clone come di sola lettura.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rox
spec:
  accessModes:
    - ReadOnlyMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
  dataSource:
    kind: PersistentVolumeClaim
    name: gcnv-san-rwo
```

ReadWriteMany (RWX) — solo raw block

ReadWriteMany è supportato solo quando `volumeMode: Block`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-raw-rwx
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

Comportamento del volume a blocchi

I volumi a blocchi vengono forniti come LUN iSCSI e presentati ai nodi Kubernetes come dispositivi a blocchi.

Volumi a blocchi:

- Utilizzare il protocollo iSCSI
- Supporta la presentazione sia del file system che dei blocchi raw
- Sono allegati e gestiti da Trident
- Supporta più modalità di accesso a Kubernetes

Modalità di accesso

I volumi a blocchi forniti da Trident supportano le seguenti modalità di accesso:

- `ReadWriteOnce` (RWO)
- `ReadOnlyMany` (ROX)
- `ReadWriteOncePod` (RWOP)
- `ReadWriteMany` (RWX), supportato solo quando `volumeMode: Block`

Comportamento di volumeMode

Il `volumeMode` campo controlla come viene esposto un volume di blocco:

- `Filesystem` Trident formatta e monta il volume.
- `Block` Trident collega il dispositivo e lo espone come raw block device.

Operazioni supportate

Volumi a blocchi forniti utilizzando il `google-cloud-netapp-volumes-san` driver support:

- Crea
- Elimina
- Clona
- Snapshot
- Ridimensionare
- Importare

Overhead delle dimensioni del volume

I volumi a blocchi di Google Cloud NetApp Volumes includono overhead interno dei metadati. Di conseguenza, la dimensione del dispositivo visibile al kernel è inferiore alla capacità di provisioning.

Overhead osservato:

- Circa 300 KiB quando crei un volume
- Fino a circa 107 MiB dopo il ridimensionamento

Google Cloud NetApp Volumes accetta allocazioni di capacità in GiB interi. Per garantire che la dimensione utilizzabile del dispositivo soddisfi o superi la richiesta del PVC, Trident:

- Arrotonda la dimensione richiesta al GiB intero successivo.
- Aggiunge un buffer da 1 GiB.

Ad esempio:

- Richiesta PVC: 100 GiB.
- Capacità allocata: 101 GiB.
- Capacità utilizzabile visibile all'applicazione: almeno 100 GiB.

Esempi di pod

Volume a blocchi montato sul filesystem (RWO)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-rwo
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeMounts:
    - name: data
      mountPath: /mnt/data
  volumes:
  - name: data
```

```
    persistentVolumeClaim:
      claimName: gcnv-san-rwo
```

Dispositivo a blocchi raw (RWX)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-raw-rwx
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeDevices:
    - name: data
      devicePath: /dev/xda
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-raw-rwx
```

Comportamento di attach e mount

Per i volumi SAN forniti da Google Cloud NetApp Volumes:

- Trident crea un Logical Unit Number (LUN) in un pool di storage Flex Unified.
- Durante la pubblicazione, Trident mappa la LUN su un gruppo host per nodo.
- Durante la messa in scena del nodo, Trident:
  - Accede all'iSCSI target
  - Scopre il LUN
  - Configura multipath
- Se `volumeMode: Filesystem`, Trident formatta il dispositivo se necessario e lo monta.
- Se `volumeMode: Block`, Trident collega il dispositivo e lo espone direttamente al pod senza formattarlo o montarlo.

> I volumi a blocchi SAN non forniscono blocco distribuito o coordinamento delle scritture. Quando un volume a blocchi è accessibile da più nodi (ReadWriteMany con `volumeMode: Block`), l'applicazione o il file system devono gestire la concorrenza.

Preparazione per la configurazione di un backend Google Cloud NetApp Volumes

Prima di poter configurare il back-end di Google Cloud NetApp Volumes, devi verificare che siano soddisfatti i seguenti requisiti.

Prerequisiti per i volumi NFS o SMB

Se utilizzi Google Cloud NetApp Volumes per la prima volta o in una nuova posizione, è necessaria una configurazione iniziale per impostare Google Cloud NetApp Volumes e creare un volume NFS o SMB. Consulta "Prima di iniziare".

Prima di configurare il back-end di Google Cloud NetApp Volumes, assicurati di disporre di quanto segue:

- Un account Google Cloud configurato con il servizio Google Cloud NetApp Volumes. Fare riferimento alla "Google Cloud NetApp Volumes".
- Numero di progetto dell'account Google Cloud. Fare riferimento alla "Identificazione dei progetti".
- Un account di servizio Google Cloud con il ruolo NetApp Volumes Admin (`roles/netapp.admin`). Fare riferimento alla "Ruoli e autorizzazioni di Identity and Access Management".
- File chiave API per il tuo account GCNV. Fare riferimento alla "Creare una chiave dell'account del servizio"
- Un pool di storage. Fare riferimento alla "Panoramica dei pool di storage".

Per ulteriori informazioni su come configurare l'accesso a Google Cloud NetApp Volumes, fare riferimento a "Configurare l'accesso a Google Cloud NetApp Volumes".

Opzioni ed esempi di configurazione di backend dei volumi Google Cloud NetApp

Scopri le opzioni di configurazione di back-end per Google Cloud NetApp Volumes e consulta gli esempi di configurazione.

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	Il valore di `storageDriverName` deve essere specificato come "google-cloud-netapp-Volumes".
`backendName`	(Facoltativo) Nome personalizzato del backend dello storage	Nome del driver + "_" + parte della chiave API
`storagePools`	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.	

Parametro	Descrizione	Predefinito
`location`	La posizione di Google Cloud in cui Trident crea volumi GCNV. Quando si creano cluster Kubernetes tra aree, i volumi creati in a `location` possono essere utilizzati nei carichi di lavoro pianificati sui nodi in più aree Google Cloud. Il traffico interregionale comporta un costo aggiuntivo.	
`apiKey`	Chiave API per l'account del servizio Google Cloud con il `netapp.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). L' `apiKey` deve includere coppie chiave-valore per le seguenti chiavi: `type`, `project_id`, , `client_email`, , `client_id` `auth_uri` `token_uri` `auth_provider_x509_cert_url`, , e `client_x509_cert_url`.	
`nfsMountOptions`	Controllo dettagliato delle opzioni di montaggio NFS.	"nfsvers=3"
`limitVolumeSize`	Il provisioning non riesce se le dimensioni del volume richiesto sono superiori a questo valore.	"" (non applicato per impostazione predefinita)
`serviceLevel`	Il livello di servizio di un pool di storage e i relativi volumi. I valori sono `flex`, `standard`, , `premium`o `extreme`.	
`labels`	Set di etichette arbitrarie formattate con JSON da applicare sui volumi	""
`network`	Rete Google Cloud usata per GCNV Volumes.	
`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
`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`
`supportedTopologies`	Rappresenta un elenco di aree e zone supportate da questo backend. Per ulteriori informazioni, fare riferimento a "Utilizzare la topologia CSI". Ad esempio: `supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a`	

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.	"0.0.0.0/0"
`snapshotDir`	Accesso a `.snapshot` directory	`true`, `false` (Il comportamento predefinito potrebbe variare. Impostare esplicitamente) "false" per NFSv3
`snapshotReserve`	Percentuale di volume riservato agli snapshot	"" (accettare l'impostazione predefinita di 0)
`unixPermissions`	Le autorizzazioni unix dei nuovi volumi (4 cifre ottali).	""

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, Trident rileva tutti i pool di storage delegati ai volumi Google Cloud NetApp nella posizione configurata e posiziona nuovi volumi in uno di tali pool in modo casuale. Poiché `nasType` viene omesso, viene applicato il `nfs` valore predefinito e il backend esegue il provisioning dei volumi NFS.

Questa configurazione è ideale quando si inizia a utilizzare Google Cloud NetApp Volumes e si stanno facendo delle prove, ma in pratica potrebbe essere necessario fornire un ambito aggiuntivo per i volumi di cui si effettua il provisioning.

Sostituisci `<id_value>` e `<key_value>` con le credenziali del tuo account di servizio.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
```

```
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

▼ **Configurazione per volumi SMB**

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    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/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

▼ **Configurazione con il filtro StoragePools**

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  storagePools:
    - premium-pool1-europe-west6
    - premium-pool2-europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

▼ **Configurazione dei pool virtuali**

Questa configurazione backend definisce più pool virtuali in un singolo file. I pool virtuali sono definiti nella `storage` sezione . Sono utili quando disponi di più pool di storage che supportano diversi livelli di servizio e vuoi creare classi di storage in Kubernetes che ne rappresentano le caratteristiche. Le etichette dei pool virtuali vengono utilizzate per differenziare i pool. Ad esempio, nell'esempio riportato di seguito `performance` vengono utilizzate etichette e `serviceLevel` tipi per differenziare i pool virtuali.

È inoltre possibile impostare alcuni valori predefiniti applicabili a tutti i pool virtuali e sovrascrivere i valori predefiniti per i singoli pool virtuali. Nell'esempio seguente, `snapshotReserve` e `exportRule` fungono da impostazioni predefinite per tutti i pool virtuali.

Per ulteriori informazioni, fare riferimento a "Pool virtuali".

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
```

```
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
        performance: standard
      serviceLevel: standard
```

▼ **Identità cloud per GKE**

Quando installi Trident con una cloud identity GKE, i backend si autenticano come workload identity e omettono il `apiKey` blocco. La cloud identity viene configurata allo stesso modo per entrambi i driver Google Cloud NetApp Volumes. L'unica differenza è il `storageDriverName` valore: usa `google-cloud-netapp-volumes` per i volumi NAS (NFS o SMB) e `google-cloud-netapp-volumes-san` per i volumi SAN (iSCSI block).

> Il supporto per l'identità cloud per il `google-cloud-netapp-volumes-san` driver richiede Trident 26.06 o versioni successive. I volumi SAN vengono forniti dai pool di storage Flex Unified, quindi imposta `serviceLevel` su `flex`.

Driver NAS (<code>google-cloud-netapp-volumes</code>)

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
```

```
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

Driver SAN (<code>google-cloud-netapp-volumes-san</code>)

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv-san
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: flex
  storagePool: pool-flex1
```

▼ **Configurazione delle topologie supportate**

Trident facilita il provisioning dei volumi per i workload in base a regioni e zone di disponibilità. Il `supportedTopologies` blocco in questa configurazione backend viene utilizzato per fornire un elenco di aree e zone per backend. I valori di regione e zona specificati qui devono corrispondere ai valori di regione e zona dalle etichette su ogni nodo del cluster Kubernetes. Queste regioni e zone rappresentano l'elenco dei valori consentiti che possono essere forniti in una classe di archiviazione. Per le classi di archiviazione che contengono un sottoinsieme delle aree e delle zone fornite in un backend, Trident crea volumi nell'area e nella zona menzionate. Per ulteriori informazioni, fare riferimento a "Utilizzare la topologia CSI".

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-a
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-b
```

Quali sono le prossime novità?

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

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

Per verificare che il backend sia stato creato correttamente, eseguire il comando seguente:

```
kubectl get tridentbackendconfig

NAME                BACKEND NAME        BACKEND UUID
PHASE   STATUS
backend-tbc-gcnv    backend-tbc-gcnv    b2fd1ff9-b234-477e-88fd-713913294f65
Bound   Success
```

Se la creazione del backend non riesce, si è verificato un errore nella configurazione del backend. È possibile descrivere il backend utilizzando il `kubectl get tridentbackendconfig <backend-name>` comando oppure 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 eliminare il backend ed eseguire nuovamente il comando create.

Definizioni delle classi di storage

Di seguito è riportata una definizione di base `StorageClass` che fa riferimento al backend riportato sopra.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

Definizioni di esempio utilizzando il `parameter.selector` campo:

L'utilizzo `parameter.selector` consente di specificare per ciascun `StorageClass` "pool virtuale" sistema utilizzato per ospitare un volume. Gli aspetti del volume saranno definiti nel pool selezionato.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
```

```
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=standard
  backendType: google-cloud-netapp-volumes
```

Per ulteriori informazioni sulle classi di archiviazione, fare riferimento a "Creare una classe di storage".

Definizioni di esempio per volumi SMB

Utilizzando `nasType`, , `node-stage-secret-name` e `node-stage-secret-namespace`, è possibile specificare un volume SMB e fornire le credenziali di Active Directory richieste. Qualsiasi utente/password di Active Directory con autorizzazioni qualsiasi/nessuna può essere utilizzato per il segreto di fase del nodo.

▼ **Configurazione di base sullo spazio dei nomi predefinito**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  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: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  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: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  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.

Esempio di definizione PVC

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc
```

Per verificare se il PVC è associato, eseguire il seguente comando:

```
kubectl get pvc gcnv-nfs-pvc

NAME          STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS AGE
gcnv-nfs-pvc  Bound    pvc-b00f2414-e229-40e6-9b16-ee03eb79a213  100Gi
RWX            gcnv-nfs-sc  1m
```

Configura il tiering automatico per Google Cloud NetApp Volumes

Il tiering automatico viene configurato tramite i parametri del backend di Trident e le annotazioni PersistentVolumeClaim durante il provisioning dei volumi. È possibile configurare il tiering automatico per Google Cloud NetApp Volumes utilizzando Trident.

Panoramica

L'auto-tiering consente a Trident di effettuare il provisioning di volumi che spostano automaticamente i dati inattivi da un livello di prestazioni a un livello di capacità. Questo riduce i costi di storage preservando le prestazioni per i dati a cui si accede di frequente.

Trident applica le impostazioni di auto-tiering solo al momento della creazione del volume. Le modifiche successive al provisioning non sono supportate in Trident 26.10.

Concetti

Auto-tiering

L'auto-tiering sposta i dati a cui si accede raramente da un livello di prestazioni a un livello di capacità in base ai modelli di accesso. Lo spostamento dei dati avviene in modo asincrono e non è immediato.

Policy di tiering

Il criterio di tiering determina se il tiering automatico è abilitato per un volume.

Sono supportate le seguenti policy: * `auto`: Abilita la suddivisione in livelli automatica in base ai modelli di accesso * `none`: Disabilita la suddivisione in livelli automatica

Giorni di raffreddamento

I giorni di raffreddamento specificano il numero minimo di giorni per cui un blocco di dati deve rimanere inattivo prima di diventare idoneo per il tiering. I giorni di raffreddamento si applicano solo quando la tiering policy è impostata su `auto`.

Modello di configurazione

Ambiti di configurazione

L'auto-tiering può essere configurato in più ambiti:

- **Ambito del pool di archiviazione** Si applica a tutti i volumi provisioned dal pool.
- **Ambito del volume** Si applica a un singolo volume tramite annotazioni PersistentVolumeClaim.

Trident determina la configurazione effettiva in base a dove è definita ciascuna impostazione.

Precedenza di configurazione

Quando la stessa impostazione è definita in più ambiti, Trident applica il seguente ordine di precedenza:

1. Annotazioni PersistentVolumeClaim
2. Configurazione del backend Trident
3. Valori predefiniti del pool di archiviazione

Le impostazioni definite a un livello di precedenza superiore sovrascrivono i valori di livello inferiore.

Funzionalità supportate in Trident 26.10

Trident 26.10 supporta le seguenti funzionalità di auto-tiering per Google Cloud NetApp Volumes:

- Abilitazione o disabilitazione del tiering automatico durante il provisioning del volume
- Definizione di una politica di tiering nella configurazione del backend Trident
- Sovrascrivere la policy di tiering e i giorni di raffreddamento per volume utilizzando le annotazioni PVC
- Configurazione dei giorni di raffreddamento per i volumi con auto-tiering abilitato

Funzionalità non supportata in Trident 26.10

Le seguenti operazioni non sono supportate:

- Modifica delle impostazioni di auto-tiering dopo la creazione del volume
- Modifica delle policy di tiering sui volumi esistenti tramite gli aggiornamenti di Kubernetes
- Applicazione delle impostazioni di auto-tiering al di fuori dei flussi di lavoro di provisioning gestiti da Trident

Parametri di configurazione del backend

I seguenti parametri controllano il comportamento dell'auto-tiering quando definiti nella configurazione del backend Trident:

Parametro	Obbligatorio	Descrizione
tieringPolicy	No	Criterio di suddivisione in livelli per volumi (`auto` or `none`)
tieringMinimumCoolingDays	No	Numero di giorni di inattività prima che i dati vengano tiered (intervallo: 2–183, predefinito: 31)

Sovrascritture a livello di volume tramite annotazioni PersistentVolumeClaim

Annotazioni supportate

Le annotazioni PersistentVolumeClaim consentono di ignorare le impostazioni di auto-tiering per volume.

Annotazione	Descrizione
`trident.netapp.io/tieringPolicy`	Sostituisce la policy di tiering per il volume

Annotazione	Descrizione
`trident.netapp.io/tieringMinimumCoolingDays`	Sostituisce il valore dei giorni di raffreddamento per il volume

Esempio: PersistentVolumeClaim con override di auto-tiering

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-tiering-pvc
  annotations:
    trident.netapp.io/tieringPolicy: auto
    trident.netapp.io/tieringMinimumCoolingDays: "45"
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: google-cloud-netapp-volumes-auto-tiering
  resources:
    requests:
      storage: 500Gi
```

Comportamento e limitazioni

Comportamento di provisioning

- Le impostazioni di auto-tiering vengono valutate e applicate solo al momento della creazione del volume.
- Trident non riconcilia la configurazione del tiering dopo il provisioning.
- I giorni di raffreddamento vengono ignorati quando la policy di tiering è impostata su `none`.

Limitazioni della piattaforma

- Il tiering automatico è supportato solo per i volumi NAS (NFS e SMB).
- I volumi a blocchi (iSCSI) non supportano l'auto-tiering.
- Il pool di storage Google Cloud NetApp Volumes deve avere la suddivisione automatica in livelli abilitata in Google Cloud.

Valori supportati

- Intervallo valido per `tieringMinimumCoolingDays`: 2 a 183
- Valore predefinito: 31

Configura i pool di storage in modalità ONTAP di Google Cloud NetApp Volumes

A partire da Trident 26.06, puoi configurare backend esistenti `ontap-san` e `ontap-nas` backend per effettuare il provisioning dello storage su Google Cloud NetApp Volumes

(GCNV) pool di storage in modalità ONTAP. Questa funzionalità è una Tech Preview.

NOTA	Questa funzionalità è una Tech Preview in Trident 26.10. Le funzionalità Tech Preview non sono supportate per l'utilizzo in produzione. La funzionalità, i campi di configurazione e la matrice supportata possono cambiare prima della disponibilità generale (GA).

Come funziona

Trident riutilizza la logica del driver esistente `ontap-san` e `ontap-nas` e instrada le chiamate REST ONTAP attraverso un endpoint proxy GCNV. Questo approccio ti permette di eseguire il provisioning e gestire lo storage su cluster ONTAP ospitati da GCNV mantenendo gli stessi flussi di lavoro operativi che usi per un backend ONTAP diretto. La modalità ONTAP non introduce un nuovo driver di storage. La abiliti per ogni backend aggiungendo un blocco di configurazione `gcnv` a un backend `ontap-san` o `ontap-nas`.

Il percorso di provisioning è il seguente:

`PersistentVolumeClaim` → Trident (`ontap-san` o `ontap-nas`) → client REST ONTAP → proxy GCNV → cluster ONTAP nel pool GCNV

Ambito supportato e non supportato

Questa anteprima tecnica supporta le seguenti funzionalità:

- Driver: `ontap-san` (iSCSI) e `ontap-nas` (NFS o SMB).
- Operazioni del ciclo di vita tramite proxy: inizializzazione del backend, creazione ed eliminazione dei volumi e relativi flussi REST ONTAP.
- Modelli di autenticazione: Workload Identity Pool (WIP), chiave dell'account di servizio e fallback alle Application Default Credentials (ADC).

I seguenti elementi sono esclusi dall'ambito di questa Tech Preview:

- I `ontap-san-economy`, `ontap-nas-economy` e `ontap-nas-flexgroup` driver.
- Il percorso di personalità ASA r2.
- Ripiego ZAPI. La modalità ONTAP usa solo ONTAP REST.

Prerequisiti

Prima di configurare un backend in modalità ONTAP, assicurati di avere quanto segue:

- Trident 26.06 o versioni successive.
- Un pool di storage GCNV ONTAP-mode nel tuo progetto e nella posizione di destinazione.
- Autorizzazioni IAM di GCP per le operazioni proxy, con ambito appropriato per il tuo ambiente.
- Una configurazione di backend che include `proxyURL`, `projectNumber`, `location` e `poolID`.
- Per i flussi di lavoro SAN iSCSI, i prerequisiti iSCSI e multipath lato nodo per la tua piattaforma. Consulta "Preparare il nodo di lavoro".

Configura il backend

A `TridentBackendConfig` richiede `credentials` di fare riferimento a un Kubernetes Secret. Per la modalità ONTAP, il secret fornisce le credenziali per l'accesso al

`proxy GCP, non ONTAP `managementLIF / username / password. A seconda del metodo di autenticazione, può contenere campi della chiave dell'account di servizio o essere vuoto. `svm`è facoltativo: ogni pool GCNV in modalità ONTAP attualmente ha un SVM e Trident lo ricava dal pool quando viene omesso.

Per l'autenticazione con chiave dell'account di servizio, usa lo stesso schema di GCNV nativo: campi non sensibili in `gcnv.apiKey`, `private_key` e `private_key_id` nel secret. Vedi "Esempi di configurazione di Google Cloud NetApp Volumes". Per l'autenticazione con Workload Identity Pool o GKE cloud identity, vedi gli esempi in Autenticazione.

backend ontap-san

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-san
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  backendName: gcnv-ontap-san
  credentials:
    name: gcnv-sa-secret
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
  svm: "<svm-name>"
```

backend ontap-nas

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: gcnv-ontap-nas
  credentials:
    name: gcnv-sa-secret
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
```

```
    location: "<pool-location>"
    poolID: "<pool-id>"
    apiKey:
      type: service_account
      project_id: "<project-id>"
      client_email: "<service-account-email>"
      client_id: "<client-id>"
      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: "<client-x509-cert-url>"
  svm: "<svm-name>"
```

Segreto

```
apiVersion: v1
kind: Secret
metadata:
  name: gcnv-sa-secret
  namespace: trident
type: Opaque
stringData:
  private_key_id: "<private-key-id>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <private-key>
    -----END PRIVATE KEY-----
```

NOTA	Non mettere `private_key` o `private_key_id` nella `TridentBackendConfig` spec: il CRD li rifiuta. Usa `ontap-san` con lo stesso `gcnv` e lo stesso modello di credenziali per storage a blocchi.

Autenticazione

Trident risolve le credenziali per l'accesso proxy nel seguente ordine:

1. Workload Identity Pool (WIP)
2. Chiave dell'account di servizio
3. Credenziali predefinite dell'applicazione (ADC)

Per questa Tech Preview, segui le seguenti procedure:

- Preferisci Workload Identity Pool dove è disponibile.
- Non incorporare chiavi private raw nei file di backend sottoposti a controllo di versione. Archivia le credenziali in un secret di Kubernetes.

- Applica all'account di servizio un ambito IAM con privilegi minimi.

Il `location` valore nel `gcnv` blocco deve corrispondere alla posizione del pool di storage: usa la zona per un pool zonale e la regione per un pool regionale.

Pool di identità dei carichi di lavoro

Usa `gcnv.wipCredential` per la federazione esplicita dell'account esterno configurata su `TridentBackendConfig`. Su OpenShift, Trident proietta il token in `/var/run/secrets/wif-token/token`; questo è l'unico percorso del token che Trident proietta per OpenShift WIF, quindi usalo per `credential_source.file`. Installa Trident con l'audience WIF come descritto in "Configura Google Cloud NetApp Volumes con Workload Identity Federation". Il `credentials` campo fa ancora riferimento a un Secret, ma il Secret non deve contenere un `private_key`.

```
apiVersion: v1
kind: Secret
metadata:
  name: gcnv-ontap-empty
  namespace: trident
type: Opaque
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-nas-wip
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: gcnv-ontap-nas-wip
  useREST: true
  nasType: nfs
  nfsMountOptions: nfsvers=4.1
  credentials:
    name: gcnv-ontap-empty
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
    wipCredential:
      type: external_account
      audience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<wif-pool-
id>/providers/<provider-id>"
      subject_token_type: "urn:ietf:params:oauth:token-type:jwt"
```

```
      token_url: "https://sts.googleapis.com/v1/token"
      service_account_impersonation_url:
"https://iamcredentials.googleapis.com/v1/projects/-
/serviceAccounts/<service-account-email>:generateAccessToken"
      credential_source:
        file: "/var/run/secrets/wif-token/token"
```

identità cloud GKE

Per GKE, configura l'identità cloud di Trident durante l'installazione come descritto in "Usa l'identità del carico di lavoro GKE". Il backend utilizza le credenziali predefinite dell'applicazione (ADC); ometti sia `gcnv.apiKey` che `gcnv.wipCredential`. Il campo `credentials`fa ancora riferimento a un Secret, ma non inserirvi un `private_key`.

```
apiVersion: v1
kind: Secret
metadata:
  name: gcnv-ontap-empty
  namespace: trident
type: Opaque
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-nas-wi
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: gcnv-ontap-nas-wi
  useREST: true
  nasType: nfs
  nfsMountOptions: nfsvers=4.1
  credentials:
    name: gcnv-ontap-empty
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
```

Comportamento di validazione e inizializzazione

Trident valida un backend in modalità ONTAP durante l'inizializzazione e si arresta immediatamente invece di inizializzarsi solo parzialmente.

L'inizializzazione del backend non riesce se manca un campo `gcnv` obbligatorio, se il driver non è supportato per la modalità ONTAP o se la risoluzione delle credenziali non riesce.

Limitazioni note

- Questa funzionalità è una Tech Preview. Il comportamento e la matrice supportata possono cambiare prima della GA.
- Il fallback ZAPI non viene utilizzato in modalità ONTAP.
- I requisiti relativi al protocollo e alla predisposizione del nodo rimangono validi. Ad esempio, i flussi di lavoro iSCSI richiedono i prerequisiti lato nodo descritti in "Preparare il nodo di lavoro".
- Le limitazioni dell'ambiente ONTAP e GCNV esistenti rimangono valide.

Aggiornamento e compatibilità

La modalità ONTAP è opzionale per ciascun backend tramite il `gcnv` blocco di configurazione:

- I backend ONTAP esistenti non GCNV non sono interessati.
- Le implementazioni miste che combinano backend ONTAP diretti e backend proxy in modalità ONTAP sono supportate tramite la configurazione a livello di backend. Configuri ciascun backend in modo indipendente.
- Per interrompere l'utilizzo di ONTAP-mode, rimuovi o sostituisci i backend interessati. Non è necessario alcuno switch globale.

Quali sono le prossime novità?

Applica il secret prima di TBC; usa `kubectl apply` e `-n trident` per allinearti agli esempi.

Configurare un backend NetApp HCI o SolidFire

Scoprite come creare e utilizzare un backend Element con l'installazione Trident.

Dettagli driver elemento

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

Il `solidfire-san` driver di archiviazione supporta le modalità di volume *file* e *block*. Per la `Filesystem` modalità volumeMode, Trident crea un volume e crea 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	Sempre "SolidFire-san"
`backendName`	Nome personalizzato o backend dello storage	"SolidFire_" + indirizzo IP di 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. 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 "Trident"
`Types`	Specifiche QoS	
`limitVolumeSize`	Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore	"" (non applicato per impostazione predefinita)
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Esempio, {"api":false, "method":true}	nullo

ATTENZIONE 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.

Trident copia le etichette presenti su un pool di storage al LUN di storage backend 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. Questa è l'unica piscina che offre prestazioni d'oro con un `Volume Type QoS` di Gold. L'ultima StorageClass (`solidfire-silver`) richiama qualsiasi pool di storage che offre prestazioni eccezionali. Trident deciderà quale pool virtuale viene selezionato e garantirà che i requisiti di storage vengano soddisfatti.

```
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

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

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

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
`ontap-san`	NVMe/TCP Fare riferimento a. Considerazioni aggiuntive su NVMe/TCP.	Blocco	RWO, ROX, RWX, RWOP	Nessun file system; dispositivo a blocchi raw
`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`

ATTENZIONE

- 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.
- Il dimensionamento automatico dei volumi ONTAP è supportato con `ontap-san` e `ontap-san-economy`. Non è supportato con i driver NAS di ONTAP. Per dettagli e limitazioni, vedi "Configura il dimensionamento automatico del volume ONTAP".

Autorizzazioni utente

Trident può essere eseguito come amministratore di ONTAP o SVM, in genere utilizzando un utente del cluster o un utente SVM `admin` o `vsadmin` un utente con un nome diverso che svolge lo stesso ruolo. Per le implementazioni di Amazon FSX per NetApp ONTAP, Trident si aspetta un'esecuzione come amministratore ONTAP o SVM, con l'utente del cluster `fsxadmin`, un `vsadmin` utente SVM o un utente con un nome diverso che abbia lo stesso ruolo. L' `fsxadmin`utente sostituisce in modo limitato l'utente amministratore del cluster.

NOTA	Se si utilizza il `limitAggregateUsage` parametro, sono necessarie le autorizzazioni di amministratore del cluster. Quando si utilizza Amazon FSX per NetApp ONTAP con Trident, il `limitAggregateUsage` parametro non funziona con `vsadmin` gli account utente e. `fsxadmin` 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

Trident supporta il protocollo non-volatile memory express (NVMe) utilizzando il `ontap-san` driver, tra cui:

- IPv6
- Snapshot e cloni di volumi NVMe
- Ridimensionamento di un volume NVMe
- Importazione di un volume NVMe creato al di fuori di Trident in modo che il suo ciclo di vita possa essere gestito da Trident
- Multipath nativo NVMe
- Arresto anomalo o anomalo dei K8s nodi (24,06)

Trident non supporta:

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

NOTA	NVMe è supportato solo con le API REST ONTAP e non con ONTAPI (ZAPI).

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, Trident richiede che almeno un aggregato sia assegnato all'SVM.

NOTA	"Sistemi ASA r2"differiscono dagli altri sistemi ONTAP (ASA, AFF e FAS) nell'implementazione del loro livello di archiviazione. Nei sistemi ASA r2, al posto degli aggregati vengono utilizzate zone di disponibilità dello storage. Fare riferimento a"questo" Articolo della Knowledge Base su come assegnare aggregati alle SVM nei sistemi ASA r2.

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

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: 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.

ATTENZIONE	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

Trident richiede le credenziali di un amministratore con ambito SVM/cluster per comunicare con il back-end ONTAP. Si consiglia di utilizzare ruoli standard predefiniti come `admin` o `vsadmin`. Ciò garantisce la compatibilità con le future versioni di ONTAP che potrebbero esporre le API delle funzioni da utilizzare nelle future versioni di Trident. È possibile creare e utilizzare un ruolo di accesso di protezione personalizzato con Trident, ma non è consigliabile.

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

YAML

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

JSON

```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.

Abilita l'autenticazione basata sul 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
```

NOTA Dopo aver eseguito questo comando, ONTAP richiede l'inserimento del certificato. Incolla

il contenuto del `k8senv.pem` file generato nel passaggio 1, quindi premi `END` per completare l'installazione.

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

NOTA 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 backend corretto indica che Trident può comunicare con il back-end ONTAP e gestire operazioni future sui volumi.

Creare un ruolo ONTAP personalizzato per Trident

Puoi creare un ruolo cluster ONTAP con Minimum Privileges in modo da non dover utilizzare il ruolo di amministratore ONTAP per eseguire le operazioni in Trident. Quando si include il nome utente in una configurazione backend Trident, Trident utilizza il ruolo del cluster ONTAP creato per eseguire le operazioni.

Per ulteriori informazioni sulla creazione di ruoli personalizzati di Trident, fare riferimento a."Generatore di ruoli personalizzati Trident"

Utilizzo della CLI di ONTAP

1. Creare un nuovo ruolo utilizzando il seguente comando:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Creare un nome utente per l'utente Trident:

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Associare il ruolo all'utente:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Utilizzo di System Manager

In Gestione sistema di ONTAP, eseguire le seguenti operazioni:

1. **Crea un ruolo personalizzato**:
   a. Per creare un ruolo personalizzato a livello di cluster, selezionare **Cluster > Impostazioni**.

      (Oppure) per creare un ruolo personalizzato a livello di SVM, selezionare **Storage > Storage VM `required SVM` > > Impostazioni > utenti e ruoli**.

   b. Selezionare l'icona a freccia (→) accanto a **utenti e ruoli**.
   c. Selezionare **+Aggiungi** in **ruoli**.
   d. Definire le regole per il ruolo e fare clic su **Salva**.
2. **Associare il ruolo all'utente Trident**: + eseguire i seguenti passaggi nella pagina **utenti e ruoli**:
   a. Selezionare icona Aggiungi **+** in **utenti**.
   b. Selezionare il nome utente richiesto e scegliere un ruolo nel menu a discesa **ruolo**.

c. Fare clic su **Save** (Salva).

Per ulteriori informazioni, fare riferimento alle pagine seguenti:

- "Ruoli personalizzati per l'amministrazione di ONTAP" o. "Definire ruoli personalizzati"
- "Lavorare con ruoli e utenti"

Autenticare le connessioni con CHAP bidirezionale

Trident può autenticare le sessioni iSCSI con CHAP bidirezionale per i `ontap-san` driver e. `ontap-san-economy` Ciò richiede l'attivazione dell' `useCHAP`opzione nella definizione di backend. Quando è impostato su `true`, Trident configura la protezione 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
```

ATTENZIONE	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

Impostando `useCHAP` su true, l'amministratore dello storage richiede a Trident di 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** non sono già presenti LUN preesistenti nel volume, Trident imposterà il tipo di protezione predefinito su `CHAP` e procederà alla configurazione dell'iniziatore CHAP e del nome utente e dei segreti di destinazione.

- Se la SVM contiene LUN, Trident non attiva il protocollo CHAP nella 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, Trident crea un CRD corrispondente `tridentbackend` e memorizza i segreti CHAP e i nomi utente come segreti Kubernetes. Tutti i PVS creati da Trident su questo backend verranno montati e collegati tramite CHAP.

Ruota le credenziali e aggiorna i backend

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.

ATTENZIONE	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 utilizzando l'interfaccia a riga di comando di ONTAP o ONTAP System Manager poiché Trident non sarà in grado di accettare 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 non subiranno alcun problema e continueranno a rimanere attive se le credenziali vengono aggiornate da Trident sulla SVM. Le nuove connessioni utilizzano le credenziali aggiornate e le connessioni esistenti continuano a rimanere attive. Disconnettendo e riconnettendo il vecchio PVS, verranno utilizzate le credenziali aggiornate.

Opzioni ed esempi di configurazione DELLA SAN ONTAP

Crea e usa i driver SAN ONTAP con la tua installazione di Trident. Questa sezione fornisce esempi di configurazione del backend e dettagli per la mappatura dei backend a StorageClasses. "Sistemi ASA r2" differiscono dagli altri sistemi ONTAP (ASA, AFF e FAS) nell'implementazione del loro livello di storage. Queste variazioni influiscono sull'uso di alcuni parametri come indicato. "Scopri di più sulle differenze tra i sistemi ASA r2 e gli altri sistemi ONTAP". Nella configurazione del backend di Trident, non devi specificare che il tuo sistema è ASA r2. Quando selezioni `ontap-san` come `storageDriverName`, Trident rileva automaticamente ASA r2 o altri sistemi ONTAP. Alcuni parametri di configurazione del backend non sono applicabili ai sistemi ASA r2 come indicato nella tabella seguente.

NOTA	Solo il `ontap-san` Il driver (con protocolli iSCSI, NVMe/TCP e FC) è supportato per i sistemi ASA r2.

Opzioni di configurazione back-end

A partire dalla versione 26.06, puoi abilitare il posizionamento bilanciato di ONTAP così che ONTAP selezioni l'aggregato per i nuovi volumi. Consulta "Usa un posizionamento bilanciato con i backend ONTAP" per i requisiti e gli esempi.

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

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di archiviazione. Non puoi modificarlo dopo la creazione. Per aggiornare questo parametro, devi creare un nuovo backend.	`ontap-san` o. `ontap-san-economy`
`backendName`	Nome personalizzato o backend dello storage	Nome del driver + "_" + dataLIF

Parametro	Descrizione	Predefinito
`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 in modo da utilizzare gli indirizzi IPv6 se 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 lo switchover di MetroCluster senza problemi, vedere la Esempio MetroCluster. **NOTA** Se stai utilizzando credenziali "vsadmin", `managementLIF` devi essere quelle della SVM; se utilizzi credenziali "admin", `managementLIF` devi essere quelle del cluster.	"10,0.0,1", "[2001:1234:abcd::fefe]"
`dataLIF`	Indirizzo IP del protocollo LIF. Può essere impostato in modo da utilizzare gli indirizzi IPv6 se 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]` . **Non specificare per iSCSI.** Trident utilizza "Mappa LUN selettiva ONTAP" per rilevare le LIF iSCSI necessarie per stabilire una sessione multi-path. Viene generato un avviso se `dataLIF` è definito esplicitamente. **Omettere per MetroCluster.** Consultare 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` for Trident per configurare e utilizzare il protocollo CHAP bidirezionale come autenticazione predefinita per la SVM fornita nel backend. Per ulteriori informazioni, fare riferimento alla "Prepararsi a configurare il backend con i driver SAN ONTAP" sezione. **Non supportato per FCP o NVMe/TCP.**	`false`
`chapInitiatorSecret`	Segreto iniziatore CHAP. Necessario se `useCHAP=true`	""

Parametro	Descrizione	Predefinito
`labels`	Set di etichette arbitrarie formattate con JSON da applicare sui volumi	""
`chapTargetInitiatorSecret`	CHAP target Initiator secret. Necessario se `useCHAP=true`	""
`chapUsername`	Nome utente inbound. Necessario se `useCHAP=true`	""
`chapTargetUsername`	Nome utente di destinazione. Necessario se `useCHAP=true`	""
`clientCertificate`	Valore del certificato client codificato con base64. Utilizzato per l'autenticazione basata su certificato	""
`clientPrivateKey`	Valore codificato in base64 della chiave privata del client. Utilizzato per l'autenticazione basata su certificato	""
`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 sulle credenziali. Per l'autenticazione di Active Directory, vedere "Autenticare Trident su un SVM backend utilizzando le credenziali di Active Directory".	""
`password`	Password necessaria per comunicare con il cluster ONTAP . Utilizzato per l'autenticazione basata sulle credenziali. Per l'autenticazione di Active Directory, vedere "Autenticare Trident su un SVM backend utilizzando le credenziali di Active Directory".	""
`svm`	Macchina virtuale per lo storage da utilizzare	Derivato se un SVM `managementLIF` è specificato
`storagePrefix`	Prefisso utilizzato durante il provisioning di nuovi volumi nell'SVM. Non può essere modificato in seguito. Per aggiornare questo parametro, devi creare un nuovo backend.	`trident`

Parametro	Descrizione	Predefinito
`aggregate`	Aggregato per il provisioning (facoltativo; se impostato, deve essere assegnato all'SVM). Per il `ontap-nas-flexgroup` driver, questa opzione viene ignorata. Se non assegnato, è possibile utilizzare uno qualsiasi degli aggregati disponibili per il provisioning di un volume FlexGroup. **NOTA** Quando l'aggregato viene aggiornato in SVM, viene aggiornato automaticamente anche in Trident tramite polling di SVM, senza dover riavviare il Trident Controller. Quando hai configurato un aggregato specifico in Trident per il provisioning dei volumi, se l'aggregato viene rinominato o spostato fuori da SVM, il backend passa allo stato di errore in Trident durante il polling dell'aggregato SVM. Devi cambiare l'aggregato con uno presente su SVM oppure rimuoverlo del tutto per riportare online il backend. **Non specificare per i sistemi ASA r2**. **Non specificare quando `useBalancedPlacement` è impostato su `true`.**	""
`useBalancedPlacement`	Delega la selezione dell'aggregato per i nuovi volumi a ONTAP [Boolean]. Richiede ONTAP 9.19.1 o versioni successive e `useREST` impostato su `true`. Non può essere combinato con `aggregate`. Consulta la documentazione "Usa un posizionamento bilanciato con i backend ONTAP" per requisiti ed esempi.	`false`
`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`. Fornito `fsxadmin` e `vsadmin` non contiene le autorizzazioni necessarie per recuperare l'utilizzo dell'aggregato e limitarlo mediante Trident. **Non specificare per i sistemi ASA r2**.	"" (non applicato per impostazione predefinita)
`limitVolumeSize`	Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore. Limita anche le dimensioni massime dei volumi che gestisce per i 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 utilizzare le API REST ONTAP . `useREST`Quando impostato su `true` , Trident utilizza le API REST ONTAP per comunicare con il backend; quando impostato su `false` Trident utilizza chiamate ONTAPI (ZAPI) per comunicare con il backend. Questa funzionalità richiede ONTAP 9.11.1 e versioni successive. Inoltre, il ruolo di accesso ONTAP utilizzato deve avere accesso a `ontapi` applicazione. Ciò è soddisfatto dal predefinito `vsadmin` E `cluster-admin` ruoli. A partire dalla versione Trident 24.06 e ONTAP 9.15.1 o successiva, `useREST` è impostato su `true` per impostazione predefinita; modifica `useREST` A `false` per utilizzare le chiamate ONTAPI (ZAPI). `useREST`è completamente qualificato per NVMe/TCP. **NOTA** NVMe è supportato solo con le API REST ONTAP e non con ONTAPI (ZAPI). **Se specificato, impostare sempre su `true` per sistemi ASA r2**.	`true` Per ONTAP 9.15.1 o versioni successive, altrimenti `false`.
`sanType`	Utilizzare per selezionare `iscsi` iSCSI, `nvme` NVMe/TCP o `fcp` SCSI over Fibre Channel (FC).	`iscsi` se vuoto

Parametro	Descrizione	Predefinito
`formatOptions`	Usa `formatOptions` per specificare gli argomenti della riga di comando per il comando `mkfs`, che viene applicato ogni volta che un volume viene formattato. Questo ti permette di formattare il volume secondo le tue preferenze. Assicurati di specificare le formatOptions in modo simile alle opzioni del comando mkfs, escludendo il percorso del dispositivo. Esempio: "-E nodiscard" **Supportato per `ontap-san` E `ontap-san-economy` driver con protocollo iSCSI. Inoltre, supportati per sistemi ASA r2 quando si utilizzano i protocolli iSCSI e NVMe/TCP.**	
`limitVolumePoolSize`	Dimensioni massime degli FlexVol richiedibili quando si utilizzano le LUN di un backend ONTAP-san-economy.	"" (non applicato per impostazione predefinita)
`denyNewVolumePools`	Limita `ontap-san-economy` i backend dalla creazione di nuovi volumi FlexVol per contenere le proprie LUN. Per il provisioning di nuovi PVS vengono utilizzati solo i FlexVol preesistenti.	

Consigli per l'uso di formatOptions

Trident consiglia le seguenti opzioni per velocizzare il processo di formattazione:

- **-E nodiscard (ext3, ext4):** Non tentare di scartare i blocchi in fase di mkfs (scartare i blocchi inizialmente è utile su dispositivi a stato solido e storage sparse/thin-provisioned). Sostituisce l'opzione obsoleta "-K" ed è applicabile ai file system ext3, ext4.
- **-K (xfs):** Non tentare di scartare blocchi al momento dell'esecuzione di mkfs. Questa opzione è applicabile al file system xfs.

Autenticare Trident su un SVM backend utilizzando le credenziali di Active Directory

È possibile configurare Trident per l'autenticazione a un SVM backend utilizzando le credenziali di Active Directory (AD). Prima che un account AD possa accedere all'SVM, è necessario configurare l'accesso del controller di dominio AD al cluster o all'SVM. Per l'amministrazione del cluster con un account AD, è necessario creare un tunnel di dominio. Fare riferimento a "Configurare l'accesso al controller di dominio Active Directory in ONTAP" per i dettagli.

passi

1. Configurare le impostazioni del Domain Name System (DNS) per un SVM backend:

```
vserver services dns create -vserver <svm_name> -dns-servers
<dns_server_ip1>,<dns_server_ip2>
```

2. Eseguire il seguente comando per creare un account computer per l'SVM in Active Directory:

```
vserver active-directory create -vserver DataSVM -account-name ADSERVER1
-domain demo.netapp.com
```

3. Utilizzare questo comando per creare un utente o un gruppo AD per gestire il cluster o SVM

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. Nel file di configurazione del backend Trident , impostare `username` E `password` parametri rispettivamente per il nome utente o gruppo AD e la password.

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" **Se specificato, impostare su `true` per sistemi ASA r2**.
`spaceReserve`	Modalità di prenotazione dello spazio; "nessuno" (sottile) o "volume" (spesso). **Impostato su `none` per sistemi ASA r2**.	"nessuno"
`snapshotPolicy`	Policy Snapshot da utilizzare. **Impostato su `none` per sistemi ASA r2**.	"nessuno"
`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 criteri QoS con Trident richiede ONTAP 9.8 o versioni successive. È necessario utilizzare un gruppo di criteri QoS non condiviso e garantire che il gruppo di criteri venga applicato singolarmente a ciascun componente. Un gruppo di policy QoS condiviso impone un 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 del volume riservato alle snapshot. **Non specificare per i sistemi ASA r2**.	"0" se `snapshotPolicy` è "nessuno", altrimenti ""
`splitOnClone`	Separare un clone dal suo padre al momento della creazione	"falso"
`encryption`	Abilita NetApp Volume Encryption (NVE) sul nuovo volume; il valore predefinito è `false`. NVE deve essere concesso in licenza e abilitato sul cluster per usare questa opzione. Se NAE è abilitato sul backend, qualsiasi volume effettuato il provisioning in Trident è abilitato per NAE. Per ulteriori informazioni, consulta: "Come funziona Trident con NVE e NAE".	"false" **Se specificato, impostare su `true` per sistemi ASA r2**.
`luksEncryption`	Attivare la crittografia LUKS. Fare riferimento alla "Utilizzo di Linux Unified Key Setup (LUKS)".	"" **Impostato su `false` per sistemi ASA r2**.

Parametro	Descrizione	Predefinito
`tieringPolicy`	Criterio di suddivisione in livelli per utilizzare "none" **Non specificare per i sistemi ASA r2**.	
`nameTemplate`	Modello per creare nomi di volume personalizzati.	""

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'
```

NOTA Per tutti i volumi creati utilizzando il `ontap-san` driver, Trident aggiunge un ulteriore 10% di capacità al FlexVol per ospitare i metadati del LUN. Il LUN viene predisposto con la dimensione esatta richiesta dall'utente nel PVC. Trident aggiunge il 10% al FlexVol (mostrato come Dimensione disponibile in ONTAP). Ora ottieni la quantità di capacità utilizzabile che hai richiesto. Questa modifica impedisce anche che i LUN diventino di sola lettura, a meno che lo spazio disponibile non sia completamente utilizzato. Questo non si applica a ontap-san-economy.

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

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

L'1,1 è il 10 percento in più Trident aggiunge al FlexVol per ospitare i metadati LUN. Per `snapshotReserve` = 5% e richiesta PVC = 5 GiB, la dimensione totale del volume è 5,79 GiB e la dimensione disponibile è 5,5 GiB .

`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.

NOTA	Se utilizzi Amazon FSX su NetApp ONTAP con Trident, NetApp consiglia di specificare i nomi DNS per le LIF invece degli 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 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` ed omette i `svm` parametri. Ad esempio:

```
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
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 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 di posizionamento bilanciato**

Questo esempio abilita il posizionamento bilanciato di ONTAP, così che ONTAP selezioni l'aggregato per i nuovi volumi. Il posizionamento bilanciato richiede ONTAP 9.19.1 o versioni successive. Consulta "Usa un posizionamento bilanciato con i backend ONTAP" per i dettagli.

```
---
version: 1
storageDriverName: ontap-san
backendName: san-balanced-placement
managementLIF: 10.0.0.1
svm: svm_iscsi
username: vsadmin
password: <password>
useBalancedPlacement: true
```

▼ **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
```

▼ **Esempio di SCSI su FC (FCP)**

Devi disporre di una SVM configurata con FC sul back-end ONTAP. Configurazione backend di base per FC.

```
---
version: 1
backendName: fcp-backend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_fc
username: vsadmin
password: password
sanType: fcp
useREST: true
```

▼ **Esempio di configurazione backend con nameTemplate**

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

▼ **Esempio di formattoOpzioni per il driver ONTAP-san-economy**

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: ""
```

```
svm: svm1
username: ""
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: -E nodiscard
```

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.

Trident imposta le etichette di provisioning nel campo "commenti". I commenti vengono impostati sulle copie FlexVol volume Trident. 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 StorageClass definizioni si riferiscono a Esempi di backend con pool virtuali. Utilizzando il campo `parameters.selector`, ciascuna StorageClass indica quali pool virtuali puoi usare per ospitare un volume. Ora il volume ha le caratteristiche definite nel pool virtuale scelto.

- Il `protection-gold` StorageClass mappa il primo pool virtuale nel `ontap-san` backend. Questo è l'unico pool che offre protezione di livello oro.

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

- La `protection-not-gold` StorageClass corrisponde al secondo e al terzo pool virtuale nel `ontap-san` backend. Questi sono gli unici pool che offrono un livello di protezione diverso da gold.

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

- La `app-mysqldb` StorageClass corrisponde al terzo pool virtuale nel `ontap-san-economy` backend. Questo è l'unico pool che offre la configurazione del pool di storage per le app 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"
```

- La `protection-silver-creditpoints-20k` StorageClass corrisponde al secondo pool virtuale nel `ontap-san` backend. Questo è l'unico pool che offre protezione di livello argento e 20000 punti 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"
```

- La `creditpoints-5k` StorageClass corrisponde al terzo pool virtuale nel `ontap-san` backend e al quarto pool virtuale nel `ontap-san-economy` backend. Queste sono le uniche offerte di pool con 5000 creditpoints.

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

- La `my-test-app-sc` StorageClass corrisponde al `testAPP` pool virtuale nel `ontap-san` driver con `sanType: nvme`. Questo è l'unico pool che offre `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"
```

Trident decide quale pool virtuale viene selezionato e garantisce che il requisito di storage sia soddisfatto.

Aggiorna i LIF dei dati NVMe

Abilita l'aggiornamento dei LIF dati NVMe per consentire ai volumi NVMe pubblicati di usare i LIF aggiunti, rimossi o con un nuovo indirizzo senza ricreare il backend o riavviare

Trident.

Panoramica

Quando aggiungi un backend SAN ONTAP, Trident rileva le LIF dati NVMe per l'SVM. Per impostazione predefinita, Trident conserva tale set di LIF finché non aggiungi nuovamente il backend o non riavvii Trident.

Imposta `enableDataLIFRefresh` su `true` per aggiornare i LIF dei dati NVMe mentre Trident è in esecuzione. I volumi NVMe pubblicati usano quindi i LIF aggiunti, rimossi o a cui è assegnato un nuovo indirizzo.

Questa opzione è un parametro di installazione, non un parametro di configurazione del backend. È disabilitata per impostazione predefinita.

La funzionalità di aggiornamento LIF dei dati NVMe è disponibile a partire da Trident 26.10.

Questa funzionalità si applica a:

- `ontap-san`con `sanType: nvme` su ONTAP unificato.
- `ontap-san` utilizzando NVMe/TCP su ASA r2.

AVVERTENZA	Rimuovere o sostituire tutti i LIF di dati contemporaneamente provoca interruzioni. Se prevedi di rimuovere o sostituire tutti i LIF di dati, assicurati prima che sia già presente un LIF sostitutivo, così non interrompi l'accesso ai dati.

Abilita l'aggiornamento delle LIF dati NVMe

Operatore Trident

Per una nuova installazione, imposta `enableDataLIFRefresh` nella `TridentOrchestrator` specifica:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableDataLIFRefresh: true
```

Applica il manifest:

```
kubectl apply -f tridentorchestrator_cr.yaml
```

Per un'installazione esistente, applica la patch `TridentOrchestrator`:

```
kubectl patch torc trident --type=merge -p
```

```
'{"spec":{"enableDataLIFRefresh":true}}'
```

Verifica il valore applicato:

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableDataLIFRefresh}'
```

Timone

Per una nuova installazione, imposta il valore Helm:

```
helm install trident netapp-trident/trident-operator \
  --namespace trident --create-namespace \
  --set enableDataLIFRefresh=true
```

Per un'installazione Helm esistente, aggiorna la release con il valore abilitato:

```
helm upgrade trident netapp-trident/trident-operator \
  --namespace trident \
  --reuse-values \
  --set enableDataLIFRefresh=true
```

Puoi anche impostare `enableDataLIFRefresh: true` in un file di valori Helm e passarlo a `helm upgrade` con `-f <values-file>`.

tridentctl

Per una nuova installazione, usa il `--enable-datalif-refresh` flag:

```
tridentctl install --enable-datalif-refresh -n trident
```

Per abilitare l'opzione su un'installazione `tridentctl` esistente, disinstalla Trident e reinstallalo con il flag. Segui le precauzioni in "Disinstallare Trident" prima di disinstallare.

```
tridentctl uninstall -n trident
tridentctl install --enable-datalif-refresh -n trident
```

NOTA Il `--enable-datalif-refresh` flag è un'opzione di installazione e non è mostrato in `tridentctl install --help`.

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

Trident fornisce 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).

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

ATTENZIONE

- 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.
- Il dimensionamento automatico del volume ONTAP non è supportato con i driver NAS di ONTAP. Per i driver SAN supportati e i dettagli di configurazione, consulta "Configura il dimensionamento automatico del volume ONTAP".

Autorizzazioni utente

Trident può essere eseguito come amministratore di ONTAP o SVM, in genere utilizzando un utente del cluster o un utente SVM `admin` o `vsadmin` un utente con un nome diverso che svolge lo stesso ruolo.

Per le implementazioni di Amazon FSX per NetApp ONTAP, Trident si aspetta un'esecuzione come amministratore ONTAP o SVM, con l'utente del cluster `fsxadmin`, un `vsadmin` utente SVM o un utente con un nome diverso che abbia lo stesso ruolo. L' `fsxadmin`utente sostituisce in modo limitato l'utente amministratore del cluster.

NOTA

Se si utilizza il `limitAggregateUsage` parametro, sono necessarie le autorizzazioni di amministratore del cluster. Quando si utilizza Amazon FSX per NetApp ONTAP con Trident, il `limitAggregateUsage` parametro non funziona con `vsadmin` gli account utente e. `fsxadmin` 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 driver ONTAP NAS. A partire dalla release 25.10, NetApp Trident supporta "Sistema di archiviazione NetApp AFX". NetApp AFX storage systems differiscono dagli altri sistemi ONTAP (ASA, AFF e FAS) nell'implementazione del loro livello di storage. Nella configurazione del backend Trident non è necessario specificare che il sistema è AFX. Quando si seleziona `ontap-nas` come `storageDriverName`, Trident rileva automaticamente i sistemi AFX.

NOTA	Solo il `ontap-nas` il driver (con protocollo NFS) è supportato per i sistemi AFX; il protocollo SMB non è supportato.

Requisiti

- Per tutti i backend ONTAP, Trident richiede che almeno un aggregato sia assegnato all'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.
- Trident supporta volumi SMB montati su pod in esecuzione solo sui nodi Windows. Per ulteriori informazioni, fare riferimento alla Preparatevi al provisioning dei volumi SMB sezione.

Autenticare il backend ONTAP

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 l'installazione di un certificato sul backend affinché 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.

ATTENZIONE	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

Trident richiede le credenziali di un amministratore con ambito SVM/cluster per comunicare con il back-end ONTAP. Si consiglia di utilizzare ruoli standard predefiniti come `admin` o `vsadmin`. Ciò garantisce la compatibilità con le future versioni di ONTAP che potrebbero esporre le API delle funzioni da utilizzare nelle future versioni di Trident. È possibile creare e utilizzare un ruolo di accesso di protezione personalizzato con Trident, ma non è consigliabile.

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

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
        "name": "secret-backend-creds"
    }
}
```

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

NOTA 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 backend corretto indica che Trident può comunicare con il back-end ONTAP e gestire operazioni future sui volumi.

Creare un ruolo ONTAP personalizzato per Trident

Puoi creare un ruolo cluster ONTAP con Minimum Privileges in modo da non dover utilizzare il ruolo di

amministratore ONTAP per eseguire le operazioni in Trident. Quando si include il nome utente in una configurazione backend Trident, Trident utilizza il ruolo del cluster ONTAP creato per eseguire le operazioni.

Per ulteriori informazioni sulla creazione di ruoli personalizzati di Trident, fare riferimento a."Generatore di ruoli personalizzati Trident"

Utilizzo della CLI di ONTAP

1. Creare un nuovo ruolo utilizzando il seguente comando:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Creare un nome utente per l'utente Trident:

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Associare il ruolo all'utente:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Utilizzo di System Manager

In Gestione sistema di ONTAP, eseguire le seguenti operazioni:

1. **Crea un ruolo personalizzato**:
   a. Per creare un ruolo personalizzato a livello di cluster, selezionare **Cluster > Impostazioni**.

      (Oppure) per creare un ruolo personalizzato a livello di SVM, selezionare **Storage > Storage VM `required SVM` > > Impostazioni > utenti e ruoli**.

   b. Selezionare l'icona a freccia (→) accanto a **utenti e ruoli**.
   c. Selezionare **+Aggiungi** in **ruoli**.
   d. Definire le regole per il ruolo e fare clic su **Salva**.
2. **Associare il ruolo all'utente Trident**: + eseguire i seguenti passaggi nella pagina **utenti e ruoli**:
   a. Selezionare icona Aggiungi **+** in **utenti**.
   b. Selezionare il nome utente richiesto e scegliere un ruolo nel menu a discesa **ruolo**.
   c. Fare clic su **Save** (Salva).

Per ulteriori informazioni, fare riferimento alle pagine seguenti:

- "Ruoli personalizzati per l'amministrazione di ONTAP" o. "Definire ruoli personalizzati"
- "Lavorare con ruoli e utenti"

Gestire le policy di esportazione NFS

Trident utilizza le policy di esportazione NFS per controllare l'accesso ai volumi forniti.

Trident fornisce due opzioni quando si utilizzano i criteri di esportazione:

- Trident è in grado di gestire in modo dinamico il criterio di esportazione; in questa modalità operativa, l'amministratore dello storage specifica un elenco di blocchi CIDR che rappresentano indirizzi IP consentiti. Trident aggiunge automaticamente al criterio di esportazione gli indirizzi IP dei nodi applicabili che rientrano in questi intervalli al momento della pubblicazione. In alternativa, quando non vengono specificate CIDR, tutti gli IP unicast con ambito globale trovati nel nodo in cui il volume pubblicato viene aggiunto al criterio di esportazione.
- Gli amministratori dello storage possono creare una policy di esportazione e aggiungere regole manualmente. Trident utilizza il criterio di esportazione predefinito, a meno che non venga specificato un nome di criterio di esportazione diverso nella configurazione.

Gestione dinamica delle policy di esportazione

Trident consente 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, ciò consente di limitare l'accesso al cluster di storage solo ai nodi di lavoro che montano volumi e hanno IP nell'intervallo specificato, supportando una gestione dettagliata e automatizzata.

NOTA	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-economy
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
  - 192.168.0.0/24
autoExportPolicy: true
```

NOTA	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 Trident.

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

- `autoExportPolicy` è impostato su `true`. In questo modo, Trident crea una policy di esportazione per ogni volume sottoposto a provisioning con questo backend per la `svm1` SVM e gestisce l'aggiunta e l'eliminazione di regole utilizzando `autoexportCIDRs` i blocchi di indirizzi. Fino al collegamento di un volume a un nodo, il volume utilizza un criterio di esportazione vuoto senza regole per impedire l'accesso indesiderato a tale volume. Quando un volume viene pubblicato in un nodo, Trident crea una policy di esportazione con lo stesso nome del qtree sottostante contenente l'IP del nodo all'interno del blocco CIDR specificato. Questi IP verranno aggiunti anche al criterio di esportazione utilizzato dal FlexVol volume padre
  - Ad esempio:
    - Backend UUUID 403b5326-8482-40dB-96d0-d83fb3f4daec
    - `autoExportPolicy` impostare su `true`
    - prefisso di memorizzazione `trident`
    - UUUID PVC a79bcf5f-7b6d-4a40-9876-e2551f159c1c
    - Il qtree denominato Trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c crea una policy di esportazione per il FlexVol Named , una policy di esportazione per il qtree Named e `trident-403b5326-8482-40db96d0-d83fb3f4daec`una policy di esportazione vuota `trident_empty` denominata `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c` nella SVM. Le regole per la policy di esportazione di FlexVol saranno un superset di regole contenute nelle policy di esportazione dei qtree. Il criterio di esportazione vuoto verrà riutilizzato da tutti i volumi non collegati.
- `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, Trident aggiunge tutti gli indirizzi unicast con ambito globale trovati nei nodi di lavoro con pubblicazioni.

In questo esempio, `192.168.0.0/24` viene fornito lo spazio degli indirizzi. Questo indica che gli IP dei nodi Kubernetes che rientrano in questo intervallo di indirizzi con pubblicazioni verranno aggiunti alla policy di esportazione creata da Trident. Quando Trident registra un nodo su cui viene eseguito, recupera gli indirizzi IP del nodo e li controlla in base ai blocchi di indirizzi forniti in. al momento della pubblicazione, dopo aver filtrato gli indirizzi `autoExportCIDRs` IP, Trident crea le regole dei criteri di esportazione per gli indirizzi IP del client per il nodo in cui viene pubblicato.

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 Trident crea o aggiorna un backend, è possibile controllare il backend utilizzando `tridentctl` o il CRD corrispondente `tridentbackend`:

```
./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 viene rimosso un nodo, Trident controlla tutte le policy di esportazione per rimuovere le regole di accesso corrispondenti al nodo. Rimuovendo questo IP nodo dalle policy di esportazione dei backend gestiti, 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, l'aggiornamento del backend con `tridentctl update backend` assicura che Trident gestisca automaticamente i criteri di esportazione. In questo modo, vengono create due nuove policy di esportazione denominate in base all'UUID e al nome del qtree del backend, quando necessario. I volumi presenti sul backend utilizzeranno i criteri di esportazione appena creati dopo essere stati smontati e montati nuovamente.

NOTA	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 attivo viene aggiornato, è necessario riavviare il pod Trident sul nodo. Trident aggiornerà quindi il criterio di esportazione per i backend che gestisce in modo da riflettere questa modifica dell'IP.

Preparatevi al provisioning dei volumi SMB

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

ATTENZIONE	Devi configurare i protocolli NFS e SMB/CIFS nella SVM per creare un `ontap-nas-economy` volume SMB per i cluster on-premise ONTAP. La mancata configurazione di uno di questi protocolli causerà un errore nella creazione del volume SMB.
NOTA	`autoExportPolicy` Non è supportato per i volumi 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 2022. Trident supporta volumi SMB montati su pod in esecuzione solo sui nodi Windows.
- Almeno un segreto Trident contenente le credenziali di Active Directory. Per generare 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, puoi facoltativamente creare una condivisione SMB oppure Trident può crearne una.

   NOTA 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
```

   NOTA 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 uno dei seguenti elementi: Il nome di una condivisione SMB creata utilizzando la console di gestione Microsoft o l'interfaccia CLI di ONTAP; un nome che consente a 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.**	""

Abilita SMB sicuro

A partire dalla versione 25.06, NetApp Trident supporta il provisioning sicuro dei volumi SMB creati utilizzando `ontap-nas` E `ontap-nas-economy` backend. Quando l'SMB sicuro è abilitato, è possibile fornire un accesso controllato alle condivisioni SMB per utenti e gruppi di utenti di Active Directory (AD) utilizzando gli elenchi di controllo di accesso (ACL).

Punti da ricordare

- Importazione `ontap-nas-economy` volumi non è supportato.
- Sono supportati solo i cloni di sola lettura per `ontap-nas-economy` volumi.
- Se Secure SMB è abilitato, Trident ignorerà la condivisione SMB menzionata nel backend.
- L'aggiornamento dell'annotazione PVC, dell'annotazione della classe di archiviazione e del campo backend non aggiorna l'ACL della condivisione SMB.
- L'ACL di condivisione SMB specificato nell'annotazione del PVC clone avrà la precedenza su quelli presenti nel PVC di origine.
- Assicurati di fornire utenti AD validi quando attivi SMB sicuro. Gli utenti non validi non verranno aggiunti all'ACL.
- Se si forniscono allo stesso utente AD nel backend, nella classe di archiviazione e nel PVC autorizzazioni diverse, la priorità delle autorizzazioni sarà: PVC, classe di archiviazione e quindi backend.
- SMB sicuro è supportato per `ontap-nas` importazioni di volumi gestiti e non applicabile alle importazioni di volumi non gestiti.

Fasi

1. Specificare adAdminUser in TridentBackendConfig come mostrato nel seguente esempio:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
```

```
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. Aggiungere l'annotazione nella classe di archiviazione.

   Aggiungere il `trident.netapp.io/smbShareAdUser` Annotazione alla classe di archiviazione per abilitare SMB sicuro senza errori. Il valore utente specificato per l'annotazione `trident.netapp.io/smbShareAdUser` dovrebbe essere uguale al nome utente specificato in `smbcreds` segreto. è `full_control` .

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1. Creare un PVC.

   L'esempio seguente crea un PVC:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
```

```
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

Opzioni ed esempi di configurazione del NAS ONTAP

Crea e utilizza i driver ONTAP NAS con la tua installazione di Trident. Questa sezione fornisce esempi di configurazione del backend e dettagli per la mappatura dei backend a StorageClasses. A partire dalla versione 25.10, NetApp Trident supporta "Sistemi di archiviazione NetApp AFX". I sistemi storage NetApp AFX differiscono dagli altri sistemi basati su ONTAP (ASA, AFF e FAS) nell'implementazione del loro livello di storage.

NOTA Solo il `ontap-nas` il driver (con protocollo NFS) è supportato per i sistemi NetApp AFX; il protocollo SMB non è supportato.

Opzioni di configurazione back-end

Nella configurazione del backend di Trident, non è necessario specificare che il sistema è un NetApp AFX storage system. Quando si seleziona `ontap-nas` come `storageDriverName`, Trident rileva automaticamente il sistema storage AFX. Alcuni parametri di configurazione del backend non sono applicabili ai sistemi storage AFX.

A partire dalla versione 26.06, puoi abilitare il posizionamento bilanciato di ONTAP così che ONTAP selezioni l'aggregato per i nuovi volumi. Il posizionamento bilanciato è richiesto per i sistemi storage AFX su ONTAP 9.19.1 o versioni successive. Consulta "Usa un posizionamento bilanciato con i backend ONTAP" per i requisiti e gli esempi.

La tabella seguente mostra le opzioni di configurazione del backend:

Parametro	Descrizione	Predefinito
`version`		Sempre 1

Parametro	Descrizione	Predefinito
`storageDriverName`	Nome del driver di storage. Non può essere modificato dopo la creazione. Per aggiornare questo parametro, devi creare un nuovo backend. **NOTA** Solo per i sistemi NetApp AFX `ontap-nas` è supportato.	`ontap-nas`, , `ontap-nas-economy` o `ontap-nas-flexgroup`
`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 in modo da utilizzare gli indirizzi IPv6 se 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 lo switchover di MetroCluster senza problemi, vedere la Esempio MetroCluster.	"10,0.0,1", "[2001:1234:abcd::fefe]"
`dataLIF`	Indirizzo IP del protocollo LIF. NetApp consiglia di specificare `dataLIF`. Se non viene fornita, Trident recupera le LIF dati dalla SVM. È possibile specificare un nome di dominio completo (FQDN) da utilizzare per le operazioni di montaggio NFS, consentendo di creare un DNS round-robin per bilanciare il carico su più LIF dati. Può essere modificato dopo l'impostazione iniziale. Fare riferimento alla . Può essere impostato in modo da utilizzare gli indirizzi IPv6 se 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]` . **Omettere per MetroCluster.** Consultare 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
`autoExportPolicy`	Abilita la creazione e l'aggiornamento automatici dei criteri di esportazione [booleano]. Utilizzando le `autoExportPolicy` opzioni e `autoExportCIDRs`, Trident può gestire automaticamente i criteri di esportazione.	falso
`autoExportCIDRs`	Elenco di CIDR per filtrare gli IP dei nodi di Kubernetes rispetto a quando `autoExportPolicy` è attivato. Utilizzando le `autoExportPolicy` opzioni e `autoExportCIDRs`, Trident può gestire automaticamente i criteri di esportazione.	["0,0.0,0/0", ":/0"]»

Parametro	Descrizione	Predefinito
`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 connettersi al cluster/SVM. Utilizzato per l'autenticazione basata sulle credenziali. Per l'autenticazione di Active Directory, vedere "Autenticare Trident su un SVM backend utilizzando le credenziali di Active Directory".	
`password`	Password per connettersi al cluster/SVM. Utilizzato per l'autenticazione basata sulle credenziali. Per l'autenticazione di Active Directory, vedere "Autenticare Trident su un SVM backend utilizzando le credenziali di Active Directory".	
`storagePrefix`	Prefisso utilizzato per il provisioning di nuovi volumi nella SVM. Non può essere aggiornato dopo l'impostazione **NOTA** Quando usi ontap-nas-economy e un storagePrefix di 24 o più caratteri, i qtree non hanno il prefisso di storage incluso, anche se è presente nel nome del volume.	"trident"

Parametro	Descrizione	Predefinito
`aggregate`	Aggregato per il provisioning (facoltativo; se impostato, deve essere assegnato all'SVM). Per il `ontap-nas-flexgroup` driver, questa opzione viene ignorata. Se non assegnato, è possibile utilizzare uno qualsiasi degli aggregati disponibili per il provisioning di un volume FlexGroup. **NOTA** Quando l'aggregato viene aggiornato in SVM, viene aggiornato automaticamente anche in Trident tramite polling di SVM, senza dover riavviare il Trident Controller. Quando hai configurato un aggregato specifico in Trident per il provisioning dei volumi, se l'aggregato viene rinominato o spostato fuori dalla SVM, il backend passa allo stato di errore in Trident durante il polling dell'aggregato SVM. Devi cambiare l'aggregato con uno presente sulla SVM oppure rimuoverlo del tutto per riportare online il backend. **Non specificare per i sistemi di archiviazione AFX.** **Non specificare quando `useBalancedPlacement` è impostato su `true`.**	""
`useBalancedPlacement`	Delega la selezione dell'aggregato per i nuovi volumi a ONTAP [Boolean]. Richiede ONTAP 9.19.1 o versioni successive e `useREST` impostato su `true`. Non può essere combinato con `aggregate`. Consulta la documentazione "Usa un posizionamento bilanciato con i backend ONTAP" per requisiti ed esempi. **Se specificato, impostalo sempre su `true` per i sistemi storage AFX su ONTAP 9.19.1 o versioni successive.**	`false`; `true` per sistemi di storage AFX su ONTAP 9.19.1 o versioni successive
`limitAggregateUsage`	Il provisioning non riesce se l'utilizzo è superiore a questa percentuale. **Non si applica ad Amazon FSx per ONTAP**. **Non specificare per i sistemi di archiviazione AFX.**	"" (non applicato per impostazione predefinita)

Parametro	Descrizione	Predefinito
FlexgroupAggregateList	Elenco di aggregati per il provisioning (facoltativo; se impostato, deve essere assegnato alla SVM). Tutti gli aggregati assegnati alla SVM vengono utilizzati per il provisioning di un volume FlexGroup. Supportato per il driver di archiviazione **ONTAP-nas-FlexGroup**. **NOTA** Quando l'elenco degli aggregati viene aggiornato in SVM, l'elenco viene aggiornato automaticamente in Trident tramite polling di SVM, senza dover riavviare il Trident Controller. Quando hai configurato un elenco specifico di aggregati in Trident per il provisioning dei volumi, se l'elenco degli aggregati viene rinominato o spostato fuori da SVM, il backend passa allo stato di errore in Trident durante il polling dell'aggregato SVM. Devi cambiare l'elenco degli aggregati con uno presente su SVM oppure rimuoverlo del tutto per riportare online il backend.	""
`limitVolumeSize`	Il provisioning non riesce se la dimensione del volume richiesto è superiore a questo valore.	"" (non applicato per impostazione predefinita)
`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. Impostando il valore su null, i volumi NFS vengono impostati di default. **Se specificato, impostare sempre su `nfs` per i sistemi di stoccaggio AFX**.	`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 vengono specificate opzioni di montaggio in una classe di storage, Trident utilizza le opzioni di montaggio specificate nel file di configurazione del backend di storage. Se non vengono specificate opzioni di montaggio nella classe di storage o nel file di configurazione, 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"

Parametro	Descrizione	Predefinito
`smbShare`	È possibile specificare uno dei seguenti elementi: Il nome di una condivisione SMB creata utilizzando la console di gestione Microsoft o l'interfaccia CLI di ONTAP; un nome che consente a 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`
`useREST`	Parametro booleano per utilizzare le API REST ONTAP . `useREST`Quando impostato su `true` , Trident utilizza le API REST ONTAP per comunicare con il backend; quando impostato su `false` Trident utilizza chiamate ONTAPI (ZAPI) per comunicare con il backend. Questa funzionalità richiede ONTAP 9.11.1 e versioni successive. Inoltre, il ruolo di accesso ONTAP utilizzato deve avere accesso a `ontapi` applicazione. Ciò è soddisfatto dal predefinito `vsadmin` E `cluster-admin` ruoli. A partire dalla versione Trident 24.06 e ONTAP 9.15.1 o successiva, `useREST` è impostato su `true` per impostazione predefinita; modifica `useREST` A `false` per utilizzare le chiamate ONTAPI (ZAPI). **Se specificato, impostare sempre su `true` per i sistemi di stoccaggio AFX**.	`true` Per ONTAP 9.15.1 o versioni successive, altrimenti `false`.
`limitVolumePoolSize`	Dimensioni FlexVol massime richiedibili quando si utilizzano Qtree nel backend ONTAP-nas-Economy.	"" (non applicato per impostazione predefinita)
`denyNewVolumePools`	Limita `ontap-nas-economy` i backend dalla creazione di nuovi volumi FlexVol per contenere i propri Qtree. Per il provisioning di nuovi PVS vengono utilizzati solo i FlexVol preesistenti.	
`adAdminUser`	Utente o gruppo di utenti amministratore di Active Directory con accesso completo alle condivisioni SMB. Utilizzare questo parametro per fornire diritti di amministratore alla condivisione SMB con controllo completo.	

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 Qtree	"vero"
`spaceReserve`	Modalità di prenotazione dello spazio; "nessuno" (sottile) o "volume" (spesso)	"nessuno"

Parametro	Descrizione	Predefinito
`snapshotPolicy`	Policy di Snapshot da utilizzare	"nessuno"
`qosPolicy`	Gruppo di criteri QoS da assegnare per i volumi creati. Scegliere tra qosPolicy o adaptiveQosPolicy per pool di storage/backend	""
`adaptiveQosPolicy`	Gruppo di criteri QoS adattivi da assegnare per i volumi creati. Scegliere tra qosPolicy o adaptiveQosPolicy per pool di storage/backend. Non supportato da ontap-nas-Economy.	""
`snapshotReserve`	Percentuale di volume riservato agli snapshot	"0" se `snapshotPolicy` è "nessuno", altrimenti ""
`splitOnClone`	Separare un clone dal suo padre al momento della creazione	"falso"
`encryption`	Abilita NetApp Volume Encryption (NVE) sul nuovo volume; il valore predefinito è `false`. NVE deve essere concesso in licenza e abilitato sul cluster per usare questa opzione. Se NAE è abilitato sul backend, qualsiasi volume effettuato il provisioning in Trident è abilitato per NAE. Per ulteriori informazioni, consulta: "Come funziona Trident con NVE e NAE".	"falso"
`tieringPolicy`	Criterio di tiering da utilizzare "nessuno"	
`unixPermissions`	Per i nuovi volumi	"777" per i volumi NFS; vuoto (non applicabile) per i volumi SMB
`snapshotDir`	Controlla l'accesso a. `.snapshot` directory	`true`, `false` (Impostato esplicitamente).
`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`.
`nameTemplate`	Modello per creare nomi di volume personalizzati.	""

NOTA L'utilizzo di gruppi di criteri QoS con Trident richiede ONTAP 9.8 o versioni successive. È necessario utilizzare un gruppo di criteri QoS non condiviso e assicurarsi che il gruppo di criteri venga applicato singolarmente a ciascun componente. Un gruppo di policy QoS condiviso impone un 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` , Trident ora utilizza un nuovo calcolo per garantire che FlexVol sia dimensionato correttamente con la percentuale snapshotReserve e PVC. Quando l'utente richiede un PVC, 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 meno spazio di quanto richiesto. Prima della versione 21.07, quando l'utente richiedeva un PVC (ad esempio, 5 GiB), con snapshotReserve al 50%, otteneva solo 2,5 GiB di spazio scrivibile. Questo perché ciò che l'utente ha richiesto è l'intero volume e `snapshotReserve` è una percentuale di quello. Con Trident 21.07, ciò che l'utente richiede è lo spazio scrivibile e Trident definisce lo `snapshotReserve` numero come percentuale del volume totale. Questo non si applica a `ontap-nas-economy` . Per vedere come funziona, vedere l'esempio seguente

Il calcolo è il seguente:

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

Per snapshotReserve = 50% e richiesta PVC = 5 GiB, la dimensione totale del volume è 5/.5 = 10 GiB e la dimensione disponibile è 5 GiB, che è ciò che l'utente ha richiesto nella richiesta PVC . `volume show` il comando dovrebbe mostrare risultati simili a questo esempio:



Per i backend esistenti da installazioni precedenti, esegui il provisioning dei volumi come spiegato sopra durante l'aggiornamento di Trident. Per i volumi che hai creato prima dell'aggiornamento, dovresti ridimensionare i volumi affinché la modifica venga rilevata. Ad esempio, un PVC da 2 GiB con `snapshotReserve=50` in precedenza generava un volume che forniva 1 GiB di spazio scrivibile. Ridimensionando il volume a 3 GiB, ad esempio, l'applicazione avrà a disposizione 3 GiB 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.

NOTA	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**

In questo esempio viene illustrato come impostare Trident in modo che utilizzi i criteri di esportazione

dinamici per creare e gestire automaticamente i criteri di esportazione. Funziona allo stesso modo per i `ontap-nas-economy` driver e `ontap-nas-flexgroup` .

```
---
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 posizionamento bilanciato**

Questo esempio abilita il posizionamento bilanciato di ONTAP, così che ONTAP selezioni l'aggregato per i nuovi volumi. Il posizionamento bilanciato richiede ONTAP 9.19.1 o versioni successive. Consulta "Usa un posizionamento bilanciato con i backend ONTAP" per i dettagli.

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-balanced-placement
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
useBalancedPlacement: true
```

▼ **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_
```

▼ **Esempio di configurazione backend con nameTemplate**

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap-nas-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

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.

Trident imposta le etichette di provisioning nel campo "commenti". I commenti sono impostati su FlexVol for `ontap-nas` o FlexGroup for `ontap-nas-flexgroup`. 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 si riferiscono a Esempi di backend con pool virtuali. Utilizzando il campo `parameters.selector`, ciascuna StorageClass indica quali pool virtuali possono essere utilizzati per ospitare un volume. Il volume ha le caratteristiche definite nel pool virtuale scelto.

- La `protection-gold` StorageClass corrisponde al primo e al secondo pool virtuale nel `ontap-nas-flexgroup` backend. Questi sono gli unici pool che offrono la protezione di livello oro.

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

```
  fsType: "ext4"
```

- La `protection-not-gold` StorageClass corrisponde al terzo e al quarto pool virtuale nel `ontap-nas-flexgroup` backend. Questi sono gli unici pool che offrono un livello di protezione diverso da gold.

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

- La `app-mysqldb` StorageClass corrisponde al quarto pool virtuale nel `ontap-nas` backend. Questo è l'unico pool che offre la configurazione del pool di storage per app 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"
```

- La `protection-silver-creditpoints-20k` StorageClass corrisponde al terzo pool virtuale nel `ontap-nas-flexgroup` backend. Questo è l'unico pool che offre protezione di livello argento e 20000 punti 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"
```

- La `creditpoints-5k` StorageClass corrisponde al terzo pool virtuale nel `ontap-nas` backend e al secondo pool virtuale nel `ontap-nas-economy` backend. Queste sono le uniche offerte di pool con 5000 creditpoints.

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

Trident decide quale pool virtuale viene selezionato e garantisce che il requisito di storage sia soddisfatto.

Aggiornare `dataLIF` dopo la configurazione iniziale

Puoi modificare la dataLIF dopo la configurazione iniziale eseguendo il seguente comando per fornire il nuovo file JSON di backend con i dati LIF aggiornati.

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

NOTA	Se sono collegati a uno o più pod, è necessario abbassare tutti i pod corrispondenti e quindi riportarli in posizione per rendere effettiva la nuova data LIF.

Esempi di SMB sicuri

Configurazione backend con driver ontap-nas

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Configurazione backend con driver ontap-nas-economy

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Configurazione backend con pool di archiviazione

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Esempio di classe di archiviazione con driver ontap-nas

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
```

```
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

NOTA Assicurati di aggiungere `annotations` Per abilitare SMB sicuro. SMB sicuro non funziona senza annotazioni, indipendentemente dalle configurazioni impostate nel Backend o nel PVC.

Esempio di classe di archiviazione con driver ontap-nas-economy

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

Esempio di PVC con un singolo utente AD

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
```

```
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

Esempio di PVC con più utenti AD

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Usa un posizionamento bilanciato con i backend ONTAP

Puoi abilitare il posizionamento bilanciato ONTAP sui backend ONTAP. Con il posizionamento bilanciato, ONTAP seleziona l'aggregate ottimale per ogni nuovo volume, così non devi gestire l'assegnazione degli aggregate in Trident.

Come funziona il posizionamento bilanciato

Per impostazione predefinita, Trident rileva gli aggregati assegnati a un SVM, tratta ciascun aggregato come un pool di storage e seleziona un aggregato quando esegue il provisioning di un volume.

Quando abiliti il posizionamento bilanciato:

- Trident riporta l'intero SVM come un singolo pool di storage invece di riportare i singoli aggregati.
- ONTAP seleziona l'aggregato per ogni nuovo volume e bilancia i volumi tra gli aggregati disponibili.
- Trident crea volumi utilizzando l'ONTAP `/application/containers` REST API.

Il posizionamento bilanciato è disponibile per i sistemi ONTAP unificati (AFF, FAS e ASA) e per i sistemi di storage NetApp AFX.

Requisiti

- Il backend deve utilizzare ONTAP 9.19.1 o versioni successive.
- Trident deve comunicare con ONTAP tramite REST. Il posizionamento bilanciato non è disponibile con ZAPI.
- La configurazione del backend non deve specificare gli aggregati. Non puoi combinare il posizionamento bilanciato con la selezione esplicita degli aggregati.

Valori predefiniti

Il valore predefinito di `useBalancedPlacement` dipende dal sistema storage:

sistema storage	Predefinito	Note
ONTAP unificato (AFF, FAS, ASA)	`false`	Trident seleziona gli aggregati a meno che tu non abiliti il posizionamento bilanciato.
NetApp AFX (ONTAP 9.19.1 o versioni successive)	`true`	È richiesto un posizionamento bilanciato. Impostare `useBalancedPlacement` su `false` causa il fallimento della convalida del backend.
NetApp AFX (versioni precedenti a ONTAP 9.19.1)	`false`	Il posizionamento bilanciato non è disponibile.

Opzione di configurazione

Parametro	Descrizione	Predefinito
`useBalancedPlacement`	Delega la selezione aggregata per i nuovi volumi a ONTAP. Imposta questa opzione alla radice della configurazione del backend o sui singoli pool virtuali.	`false`; `true` per AFX su ONTAP 9.19.1 o versioni successive

Posizionamento bilanciato con pool virtuali

Il posizionamento bilanciato funziona con o senza pool virtuali:

- Senza pool virtuali, Trident segnala l'SVM come un singolo pool di storage e crea tutti i nuovi volumi utilizzando un posizionamento bilanciato.
- Con i pool virtuali, Trident segnala i pool virtuali e associa i volumi ad essi come di consueto. Se il pool virtuale corrispondente ha il posizionamento bilanciato abilitato, ONTAP seleziona l'aggregato per il volume.

Imposta `useBalancedPlacement` nella directory principale della configurazione del backend per applicarlo a tutti i pool virtuali, oppure impostalo su singoli pool virtuali per abilitarlo in modo selettivo.

Esempi

Il seguente esempio abilita il posizionamento bilanciato su un backend ONTAP unificato:

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-balanced-placement
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: cluster-admin
password: <password>
useBalancedPlacement: true
```

L'esempio seguente consente un posizionamento bilanciato utilizzando `TridentBackendConfig`:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: nas-balanced-placement
  managementLIF: 10.0.0.1
  svm: trident_svm
  useBalancedPlacement: true
  credentials:
    name: backend-tbc-ontap-nas-secret
```

Il seguente esempio abilita il posizionamento bilanciato solo sul `gold` pool virtuale. Trident continua a selezionare gli aggregati per i volumi che corrispondono al `bronze` pool virtuale:

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-virtual-pools
managementLIF: 10.0.0.1
svm: trident_svm
username: cluster-admin
password: <password>
storage:
- labels:
    performance: gold
  useBalancedPlacement: true
- labels:
    performance: bronze
```

Errori di convalida del backend

La creazione o l'aggiornamento del backend non riesce nei seguenti casi:

Configurazione	Risultato
`useBalancedPlacement: true` su ONTAP precedente alla versione 9.19.1	La convalida non è riuscita. Aggiorna a ONTAP 9.19.1 o versione successiva.
`useBalancedPlacement: true` su un backend utilizzando ZAPI	La convalida non è riuscita. Configura il backend per usare REST.
`useBalancedPlacement: true` combinato con aggregati specifici	La convalida non è riuscita. Rimuovi la configurazione dell'aggregate o disabilita il posizionamento bilanciato.
`useBalancedPlacement: false`su AFX con ONTAP 9.19.1 o versioni successive	La convalida non è riuscita. Per AFX su ONTAP 9.19.1 o versioni successive è richiesto un posizionamento bilanciato.

Amazon FSX per NetApp ONTAP

USA Trident con Amazon FSX per NetApp ONTAP

"Amazon FSX per NetApp ONTAP" è un servizio AWS completamente gestito che esegue file system basati sul sistema operativo per lo storage NetApp ONTAP. Fornisce funzionalità, prestazioni e amministrazione ONTAP con la scalabilità e la semplicità operativa di AWS. Un file system è la risorsa primaria in Amazon FSx ed è analogo a un cluster ONTAP on-premises. Ogni file system contiene una o più macchine virtuali di storage (SVM) e ogni SVM contiene uno o più volumi che archiviano file e directory. Questa integrazione consente ai cluster Kubernetes in esecuzione su Amazon Elastic Kubernetes Service (EKS) di eseguire il provisioning di volumi persistenti basati su ONTAP per carichi di lavoro a blocchi e file.

Requisiti

Oltre a "Requisiti Trident", per integrare FSX for ONTAP con Trident, hai bisogno di:

- 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".

> **NOTA** 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.

Considerazioni

- Volumi SMB:
  - I volumi SMB sono supportati utilizzando `ontap-nas` solo driver.
  - I volumi SMB non sono supportati con i componenti aggiuntivi Trident EKS.
  - Trident supporta volumi SMB montati solo su pod in esecuzione su nodi Windows. Consultare "Preparatevi al provisioning dei volumi SMB" per i dettagli.
- Prima di Trident 24,02, Trident non ha potuto eliminare i volumi creati su file system Amazon FSX con backup automatici abilitati. Per evitare questo problema in Trident 24,02 o versioni successive, specificare `fsxFilesystemID` , AWS , AWS `apiRegion` `apikey` e AWS `secretKey` nel file di configurazione backend per AWS FSX for ONTAP.

> **NOTA** Se si specifica un ruolo IAM in Trident, è possibile omettere esplicitamente i `apiRegion` campi , `apiKey` e `secretKey` in Trident. Per ulteriori informazioni, fare riferimento a "FSX per le opzioni di configurazione e gli esempi di ONTAP".

Utilizzo simultaneo del driver Trident SAN/iSCSI ed EBS-CSI

Se si prevede di utilizzare i driver ontap-san (ad esempio, iSCSI) con AWS (EKS, ROSA, EC2 o qualsiasi altra istanza), la configurazione multipath richiesta sui nodi potrebbe entrare in conflitto con il driver CSI di Amazon Elastic Block Store (EBS). Per garantire che il multipathing funzioni senza interferire con i dischi EBS sullo stesso nodo, è necessario escludere EBS dalla configurazione del multipathing. Questo esempio mostra un `multipath.conf` file che include le impostazioni Trident richieste escludendo i dischi EBS dal multipathing:

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

Autenticazione

Trident offre due modalità di autenticazione.

- Basato su credenziali (consigliato): Memorizza le credenziali in modo sicuro in AWS Secrets Manager. Puoi utilizzare l' `fsxadmin` utente per il tuo file system o quello `vsadmin` configurato per la tua SVM.

ATTENZIONE	Trident si aspetta di essere eseguito come `vsadmin` utente SVM o come utente con un nome diverso che abbia lo stesso ruolo. Amazon FSX per NetApp ONTAP include un `fsxadmin` utente che sostituisce in modo limitato l'utente del cluster ONTAP `admin`. Si consiglia vivamente di utilizzare `vsadmin` con Trident.

- Basato su certificato: Trident comunica con la SVM sul file system FSX utilizzando un certificato installato nella SVM.

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

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

Immagini Amazon Machine testate (AMI)

Il cluster EKS supporta vari sistemi operativi, ma AWS ha ottimizzato alcuni Amazon Machine Images (AMI) per container ed EKS. Le seguenti AMI sono state testate con NetApp Trident 25.02.

AMI	NAS	Economia NAS	ISCSI	iSCSI-economy
AL2023_x86_64_STANDARD	Sì	Sì	Sì	Sì
AL2_x86_64	Sì	Sì	Sì*	Sì*
BOTTLEROCKET_x86_64	Sì**	Sì	N/A.	N/A.
AL2023_ARM_64_STANDARD	Sì	Sì	Sì	Sì
AL2_ARM_64	Sì	Sì	Sì*	Sì*
BOTTLEROCKET_ARM_64	Sì**	Sì	N/A.	N/A.

- * Impossibile eliminare il PV senza riavviare il nodo
- ** Non funziona con NFSv3 con Trident versione 25.02.

NOTA	Se il vostro AMI desiderato non è elencato qui, non significa che non è supportato; significa semplicemente che non è stato testato. Questo elenco serve da guida per le AMI di cui è noto il funzionamento.

Prove eseguite con:

- Versione EKS: 1.32

- Metodo di installazione: Helm 25.06 e come componente aggiuntivo AWS 25.06
- Per le NAS sono stati testati sia NFSv3 che NFSv4,1.
- Per SAN è stato testato solo iSCSI, non NVMe-of.

Prove eseguite:

- Creare: Classe di archiviazione, pvc, pod
- Eliminazione: Pod, pvc (normale, qtree/lun – economia, NAS con backup AWS)

Trova ulteriori informazioni

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

Creare un ruolo IAM e un segreto AWS

Puoi configurare i pod Kubernetes in modo che accedano alle risorse AWS autenticandosi come ruolo AWS IAM invece di fornire credenziali AWS esplicite.

NOTA	Per eseguire l'autenticazione usando un ruolo AWS IAM, devi disporre di un cluster Kubernetes implementato utilizzando EKS.

Crea un segreto per AWS Secrets Manager

Poiché Trident emetterà API su un vserver FSX per gestire lo storage in modo automatico, saranno necessarie le credenziali per farlo. Il modo sicuro per passare queste credenziali è tramite un segreto di AWS Secrets Manager. Pertanto, se non ne hai già uno, dovrai creare un segreto di AWS Secrets Manager che contenga le credenziali per l'account vsadmin.

Questo esempio crea un segreto di Gestore segreti AWS per memorizzare le credenziali Trident CSI:

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

Crea criterio IAM

Trident necessita anche delle autorizzazioni AWS per funzionare correttamente. Pertanto, è necessario creare un criterio che fornisca a Trident le autorizzazioni necessarie.

I seguenti esempi creano una policy IAM utilizzando l'interfaccia a riga di comando di AWS:

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy
-document file://policy.json
  --description "This policy grants access to Trident CSI to FSxN and
Secrets manager"
```

Policy JSON esempio:

```
{
  "Statement": [
    {
      "Action": [
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes",
        "fsx:CreateVolume",
        "fsx:RestoreVolumeFromSnapshot",
        "fsx:DescribeStorageVirtualMachines",
        "fsx:UntagResource",
        "fsx:UpdateVolume",
        "fsx:TagResource",
        "fsx:DeleteVolume"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "secretsmanager:GetSecretValue",
      "Effect": "Allow",
      "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-
id>:secret:<aws-secret-manager-name>*"
    }
  ],
  "Version": "2012-10-17"
}
```

Crea l'identità del pod o il ruolo IAM per l'associazione dell'account di servizio (IRSA)

È possibile configurare un account di servizio Kubernetes per assumere un ruolo AWS Identity and Access Management (IAM) con EKS Pod Identity o un ruolo IAM per l'associazione dell'account di servizio (IRSA). Tutti i Pod configurati per utilizzare l'account di servizio possono quindi accedere a qualsiasi servizio AWS per il quale il ruolo dispone delle autorizzazioni di accesso.

Identità del pod

Le associazioni di identità dei pod Amazon EKS offrono la possibilità di gestire le credenziali per le applicazioni, in modo simile a come i profili delle istanze Amazon EC2 forniscono le credenziali alle istanze Amazon EC2.

Installa Pod Identity sul tuo cluster EKS:

Puoi creare l'identità del Pod tramite la console AWS o utilizzando il seguente comando AWS CLI:

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
```

```
eks-pod-identity-agent
```

Per maggiori informazioni fare riferimento a "Configurare l'agente di identità del pod Amazon EKS" .

Crea trust-relationship.json:

Crea trust-relationship.json per consentire al Service Principal EKS di assumere questo ruolo per l'identità del Pod. Quindi crea un ruolo con questa policy di attendibilità:

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

file trust-relationship.json:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Associare la policy del ruolo al ruolo IAM:

Associa il criterio di ruolo del passaggio precedente al ruolo IAM creato:

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

Crea un'associazione di identità pod:

Crea un'associazione di identità pod tra il ruolo IAM e l'account del servizio Trident (trident-controller)

```
aws eks create-pod-identity-association \
  --cluster-name <EKS_CLUSTER_NAME> \
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \
  --namespace trident --service-account trident-controller
```

Ruolo IAM per l'associazione dell'account di servizio (IRSA)
Utilizzando l'AWS CLI:

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

file trust-relation.json:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

Aggiornare i seguenti valori nel `trust-relationship.json` file:

- **<account_id>** - il tuo ID account AWS
- **<oidc_provider>** - l'OIDC del tuo cluster EKS. È possibile ottenere oidc_provider eseguendo:

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

Associare il ruolo IAM alla policy IAM:

Una volta creato il ruolo, allegare il criterio (creato nel passaggio precedente) al ruolo utilizzando questo comando:

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

Verificare che il provider OICD sia associato:

Verifica che il tuo provider OIDC sia associato al cluster. È possibile verificarlo utilizzando il seguente comando:

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Se l'output è vuoto, utilizzare il seguente comando per associare IAM OIDC al cluster:

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

Se si utilizza eksctl, utilizzare il seguente esempio per creare un ruolo IAM per l'account di servizio in EKS:

```
eksctl create iamserviceaccount --name trident-controller --namespace
trident \
  --cluster <my-cluster> --role-name AmazonEKS_FSxN_CSI_DriverRole
--role-only \
  --attach-policy-arn <IAM-Policy ARN> --approve
```

Installare Trident

Trident semplifica la gestione dello storage Amazon FSx for NetApp ONTAP in Kubernetes, consentendo a sviluppatori e amministratori di concentrarsi sulla distribuzione delle applicazioni. Puoi installare Trident utilizzando uno dei seguenti metodi:

- Timone
- Componente aggiuntivo EKS

Se si desidera utilizzare la funzionalità snapshot, installare il componente aggiuntivo del controller snapshot CSI. Per ulteriori informazioni, fare riferimento "Attiva la funzionalità snapshot per volumi CSI" a.

Identità del pod

1. Aggiungere il repository Trident Helm:

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

2. Installa Trident utilizzando il seguente esempio:

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 --namespace trident --create-namespace
```

È possibile utilizzare il `helm list` comando per esaminare i dettagli dell'installazione come nome, spazio dei nomi, grafico, stato, versione dell'app e numero di revisione.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2502.0    25.02.0
```

Associazione dell'account di servizio (IRSA)

1. Aggiungere il repository Trident Helm:

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

2. Imposta i valori per **cloud provider** e **cloud identity**:

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

È possibile utilizzare il `helm list` comando per esaminare i dettagli dell'installazione come nome, spazio dei nomi, grafico, stato, versione dell'app e numero di revisione.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2510.0    25.10.0
```

NOTA

Se prevedi di utilizzare iSCSI, assicurati che iSCSI sia abilitato sul computer client. Se utilizzi il sistema operativo AL2023 Worker node, puoi automatizzare l'installazione del client iSCSI aggiungendo il parametro node prep nell'installazione di helm:

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1 --namespace trident --create-namespace --
set nodePrep={iscsi}
```

Installare Trident tramite il componente aggiuntivo EKS

Il componente aggiuntivo 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 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

Attiva il componente aggiuntivo Trident per AWS

Console di gestione

1. Aprire la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home#/clusters.
2. Nel riquadro di spostamento di sinistra, selezionare **Cluster**.
3. Selezionare il nome del cluster per il quale si desidera configurare il componente aggiuntivo NetApp Trident CSI.
4. Selezionare **componenti aggiuntivi**, quindi selezionare **Ottieni altri componenti aggiuntivi**.
5. Per selezionare il componente aggiuntivo, segui questi passaggi:
   a. Scorri verso il basso fino alla sezione **Componenti aggiuntivi di AWS Marketplace** e digita **"Trident"** nella casella di ricerca.
   b. Selezionare la casella di controllo nell'angolo in alto a destra della casella Trident by NetApp.
   c. Selezionare **Avanti**.
6. Nella pagina Impostazioni **Configura componenti aggiuntivi selezionati**, effettuare le seguenti operazioni:

   NOTA **Salta questi passaggi se utilizzi l'associazione Pod Identity.**

   a. Selezionare la **versione** che si desidera utilizzare.
   b. Se si utilizza l'autenticazione IRSA, assicurarsi di impostare i valori di configurazione disponibili nelle impostazioni di configurazione facoltative:
      - Selezionare la **versione** che si desidera utilizzare.
      - Segui lo **schema di configurazione del componente aggiuntivo** e imposta il parametro **configurationValues** nella sezione **Valori di configurazione** sul role-arn creato nel passaggio precedente (il valore deve essere nel seguente formato):

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

   +
   Se si seleziona Sovrascrivi per il metodo di risoluzione dei conflitti, una o più impostazioni per il componente aggiuntivo esistente possono essere sovrascritte con le impostazioni del componente aggiuntivo Amazon EKS. Se non si attiva questa opzione e si verifica un conflitto con le impostazioni esistenti, l'operazione non riesce. È possibile utilizzare il messaggio di errore risultante per risolvere il conflitto. Prima di selezionare questa opzione, assicurati che il componente aggiuntivo Amazon EKS non gestisca le impostazioni da gestire in autonomia.
7. Scegliere **Avanti**.
8. Nella pagina **Rivedi e Aggiungi**, scegliere **Crea**.

   Al termine dell'installazione del componente aggiuntivo, viene visualizzato il componente aggiuntivo installato.

CLI AWS

1. Crea il `add-on.json` file:

Per l'identità del pod, utilizzare il seguente formato:

NOTA Utilizzare il

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

Per l'autenticazione IRSA, utilizzare il seguente formato:

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

NOTA Sostituire `<role ARN>` con l'ARN del ruolo creato nel passaggio precedente.

2. Installa il componente aggiuntivo Trident EKS.

```
aws eks create-addon --cli-input-json file://add-on.json
```

eksctl

Il seguente comando di esempio installa il componente aggiuntivo Trident EKS:

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

Aggiornare il componente aggiuntivo Trident EKS

Console di gestione

1. Aprire la console Amazon EKS https://console.aws.amazon.com/eks/home#/clusters.
2. Nel riquadro di spostamento di sinistra, selezionare **Cluster**.
3. Selezionare il nome del cluster per il quale si desidera aggiornare il componente aggiuntivo NetApp Trident CSI.
4. Selezionare la scheda **componenti aggiuntivi**.
5. Selezionare **Trident by NetApp**, quindi selezionare **Modifica**.
6. Nella pagina **Configure Trident by** (Configura server tramite NetApp*), procedere come segue:
   a. Selezionare la **versione** che si desidera utilizzare.
   b. Espandere le **impostazioni di configurazione opzionali** e modificarle secondo necessità.
   c. Selezionare **Save Changes** (Salva modifiche).

CLI AWS

Nell'esempio seguente viene aggiornato il componente aggiuntivo EKS:

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

eksctl

- Controllare la versione corrente del componente aggiuntivo FSxN Trident CSI. Sostituire `my-cluster` con il nome del cluster.

  ```
  eksctl get addon --name netapp_trident-operator --cluster my-cluster
  ```

  Esempio di output:

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v25.6.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- Aggiornare il componente aggiuntivo alla versione restituita in AGGIORNAMENTO DISPONIBILE nell'output del passaggio precedente.

  ```
  eksctl update addon --name netapp_trident-operator --version
  v25.6.0-eksbuild.1 --cluster my-cluster --force
  ```

Se si rimuove l' `--force` opzione e una delle impostazioni del componente aggiuntivo Amazon EKS è in conflitto con le impostazioni esistenti, l'aggiornamento del componente aggiuntivo Amazon EKS non viene eseguito correttamente; viene visualizzato un messaggio di errore che aiuta a risolvere il conflitto. Prima di specificare questa opzione, assicurati che il componente aggiuntivo Amazon EKS non gestisca le impostazioni da gestire, perché queste impostazioni vengono sovrascritte con questa opzione. Per ulteriori informazioni sulle altre opzioni per questa impostazione, vedere "Componenti aggiuntivi". Per ulteriori informazioni su Amazon EKS Kubernetes Field management, consulta "Gestione sul campo di Kubernetes".

Disinstallare/rimuovere il componente aggiuntivo Trident EKS

Hai due opzioni per rimuovere un add-on Amazon EKS:

- **Mantieni il software aggiuntivo sul tuo cluster** – questa opzione rimuove la gestione Amazon EKS di qualsiasi impostazione. Inoltre, rimuove la possibilità per Amazon EKS di informarti degli aggiornamenti e di aggiornare automaticamente il componente aggiuntivo Amazon EKS dopo l'avvio di un aggiornamento. Tuttavia, mantiene il software add-on sul cluster. Questa opzione rende il componente aggiuntivo un'installazione a gestione autonoma, piuttosto che un componente aggiuntivo Amazon EKS. Con questa opzione, il componente aggiuntivo non presenta tempi di inattività. Mantenere l' `--preserve` opzione nel comando per mantenere il componente aggiuntivo.
- **Rimozione del software aggiuntivo interamente dal cluster** – NetApp consiglia di rimuovere il componente aggiuntivo Amazon EKS dal cluster solo se non sono presenti risorse del cluster che dipendono da esso. Rimuovere l' `--preserve` opzione dal `delete` comando per rimuovere il componente aggiuntivo.

NOTA Se al componente aggiuntivo è associato un account IAM, l'account IAM non viene rimosso.

Console di gestione

1. Aprire la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home#/clusters.
2. Nel riquadro di spostamento di sinistra, selezionare **cluster**.
3. Selezionare il nome del cluster per il quale si desidera rimuovere il componente aggiuntivo NetApp Trident CSI.
4. Selezionare la scheda **componenti aggiuntivi**, quindi selezionare **Trident by NetApp**.*
5. Selezionare **Rimuovi**.
6. Nella finestra di dialogo **Rimuovi conferma netapp_trident-operator**, esegui quanto segue:
   a. Se si desidera che Amazon EKS smetta di gestire le impostazioni del componente aggiuntivo, selezionare **conserva su cluster**. Questa operazione consente di conservare il software aggiuntivo nel cluster in modo da poter gestire da soli tutte le impostazioni del componente aggiuntivo.
   b. Immettere **netapp_trident-operator**.
   c. Selezionare **Rimuovi**.

CLI AWS

Sostituisci `my-cluster` con il nome del cluster ed esegui il seguente comando.

```
aws eks delete-addon --cluster-name my-cluster --addon-name
```

```
netapp_trident-operator --preserve
```

eksctl

Il seguente comando disinstalla il componente aggiuntivo Trident EKS:

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

Configurare una classe di storage

Il "Oggetto Kubernetes StorageClass" identifica un provisioner e istruisce il provisioner su come effettuare il provisioning dei volumi. Questa sezione mostra come configurare un oggetto Kubernetes StorageClass che specifica Trident come provisioner.

Crea un oggetto StorageClass

Quando si crea una StorageClass per FSx for ONTAP, Trident creerà automaticamente la configurazione del backend.

NOTA Se desideri configurare manualmente il backend di storage, consulta la Crea un Kubernetes StorageClass senza configurazione automatica del backend sezione per sapere come creare separatamente il backend Trident e la storage class.

Specificare i parametri richiesti StorageClass

I seguenti tre parametri devono essere definiti durante la creazione di un StorageClass:

Parametro	Obbligatorio	Tipo	Descrizione
`fsxFilesystemID`	Sì	stringa	ID del file system FSx for NetApp ONTAP
`storageDriverName`	Sì	stringa	Driver di storage Trident (ad esempio, `ontap-nas` o `ontap-san`)
`credentialsName`	Sì	stringa	Nome del segreto Kubernetes che contiene le credenziali FSx for ONTAP

Specificare parametri facoltativi

È possibile passare parametri backend opzionali tramite la StorageClass. Definire tutti i valori opzionali come stringhe nella sezione StorageClass `parameters`. Per un elenco completo dei parametri backend, vedere: "Configurazione del backend FSx for NetApp ONTAP".

Esempi di file di configurazione StorageClass.

L'esempio seguente mostra un StorageClass che attiva la configurazione automatica del backend.

YAML

```
apiVersion: storage.k8s.io/v1
```

```
kind: StorageClass
metadata:
  name: ontap-fsx-demo
  annotations:
    description: "Demo StorageClass for FSx for NetApp ONTAP"
provisioner: csi.trident.netapp.io
parameters:
  fsxFilesystemID: "fs-0abc123"
  storageDriverName: "ontap-nas"
  credentialsName: trident-fsx-credentials
allowVolumeExpansion: true
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

JSON

```
{
  "apiVersion": "storage.k8s.io/v1",
  "kind": "StorageClass",
  "metadata": {
    "name": "ontap-fsx-demo",
    "annotations": {
      "description": "Demo StorageClass for FSx for NetApp ONTAP"
    }
  },
  "provisioner": "csi.trident.netapp.io",
  "parameters": {
    "fsxFilesystemID": "fs-0abc123",
    "storageDriverName": "ontap-nas",
    "credentialsName": "trident-fsx-credentials"
  },
  "allowVolumeExpansion": true,
  "reclaimPolicy": "Delete",
  "volumeBindingMode": "Immediate"
}
```

Crea il StorageClass

Una volta creato il file di configurazione, esegui il seguente comando per creare la storage class.

```
kubectl create -f storage-class-ontapnas.yaml
```

Ora dovresti vedere una classe storage **Basic-csi** in Kubernetes e Trident, e Trident dovrebbe aver scoperto i pool nel back-end.

```
kubectl get sc basic-csi
```

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

Dopo aver applicato il StorageClass, Trident crea automaticamente il backend. Puoi quindi creare PersistentVolumeClaims che fanno riferimento a questo StorageClass.

Verifica lo stato della configurazione del backend

Trident registra il risultato della creazione del backend nelle annotazioni di StorageClass.

Annotazione	Descrizione
`trident.netapp.io/configuratorStatus`	Risultato della configurazione (`Success` o `Failure`)
`trident.netapp.io/configuratorMessage`	Messaggio di stato o errore dettagliato
`trident.netapp.io/configuratorName`	Nome della risorsa del configuratore interno
`trident.netapp.io/managed`	Indica che la StorageClass è gestita da Trident
`trident.netapp.io/additionalStoragePools`	Pool di storage creati per questo backend

Per verificare lo stato, eseguire:

```
kubectl get storageclass ontap-fsx-demo -o yaml
```

Confermare che `trident.netapp.io/configuratorStatus` è impostato su `Success`. Se il valore è `Failure`, esaminare `trident.netapp.io/configuratorMessage` per l'errore.

Aggiungi file system FSxN aggiuntivi

Se hai bisogno di ulteriore capacità di archiviazione continuando a utilizzare lo stesso StorageClass, aggiungi ulteriori ID del file system FSxN.

Modifica la StorageClass e aggiungi la seguente annotazione:

```
metadata:
  annotations:
    trident.netapp.io/additionalFsxNFileSystemID: '["fs-
xxxxxxxxxxxxxxxxx"]'
```

Dopo aver applicato la modifica, Trident aggiorna la configurazione del backend e aggiorna le annotazioni StorageClass.

Considerazioni operative e limitazioni

- L'eliminazione di un StorageClass che ha la configurazione automatica del backend solitamente elimina il backend Trident associato. Ciò può interrompere la connettività di storage e interrompere i carichi di lavoro in esecuzione. Verifica l'impatto prima di eliminare un StorageClass gestito.
- La configurazione automatica del backend è supportata solo per AWS FSx per NetApp ONTAP.

Crea un Kubernetes StorageClass senza configurazione automatica del backend

Se desideri creare il backend Trident e la StorageClass separatamente, segui questi passaggi.

Scopri come funziona la configurazione automatica del backend

Trident deriva la configurazione del backend dalla definizione di StorageClass. Quando si applica la StorageClass, Trident convalida i parametri richiesti, crea il backend e annota la StorageClass con lo stato.

Trident crea il VolumeSnapshotClass solo una volta. Trident riutilizza lo stesso VolumeSnapshotClass per i successivi StorageClasses.

Crea il backend Trident

Per creare un backend Trident, è necessario creare un file di configurazione in formato JSON o YAML. Il file deve specificare il tipo di storage desiderato (NAS o SAN), il file system, la SVM da cui ottenerlo e come autenticarsi con essa. L'esempio seguente mostra come definire uno storage basato su NAS e utilizzare un segreto AWS per memorizzare le credenziali per la SVM che si desidera utilizzare:

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  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"
    "namespace": "trident"
  },
  "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"
    }
  }
}
```

Dettagli del driver FSX per ONTAP

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

Nome driver	Descrizione
`ontap-san`	Ogni PV di cui viene effettuato il provisioning è una LUN all'interno del proprio volume Amazon FSx for NetApp ONTAP. Consigliato per storage a blocchi.
`ontap-nas`	Ogni PV fornito è un volume Amazon FSx for NetApp ONTAP completo. Consigliato per NFS e SMB.
`ontap-san-economy`	Ogni PV di cui viene effettuato il provisioning è un LUN con un numero configurabile di LUN per Amazon FSx for NetApp ONTAP volume.
`ontap-nas-economy`	Ogni PV di cui viene effettuato il provisioning è un qtree, con un numero configurabile di qtree per Amazon FSx for NetApp ONTAP volume.
`ontap-nas-flexgroup`	Ogni PV fornito è un volume Amazon FSx for NetApp ONTAP FlexGroup completo.

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

Creare il backend

Dopo aver creato il file di configurazione, eseguire i seguenti comandi per creare e convalidare la Trident Backend Configuration (TBC):

- Creare la configurazione back-end Trident (TBC) dal file yaml ed eseguire il comando seguente:

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- Verificare che la configurazione back-end Trident (TBC) sia stata creata correttamente:

```
Kubectl get tbc -n trident
```

```
NAME                          BACKEND NAME          BACKEND UUID
PHASE   STATUS

backend-tbc-ontap-nas         tbc-ontap-nas         933e0071-66ce-4324-
b9ff-f96d916ac5e9   Bound   Success
```

Per ulteriori informazioni su altre opzioni di configurazione, consultare la Configurazione avanzata del backend ed esempi sezione seguente.

Configura una Storage Class senza configurazione automatica del backend

Di seguito sono riportati alcuni esempi di configurazioni di Storage Class da utilizzare con Trident e FSx for ONTAP.

Classe di archiviazione per NFS

È possibile utilizzare questo esempio per configurare StorageClass per volumi tramite NFS (fare riferimento alla sezione Attributi Trident di seguito per l'elenco completo degli attributi):

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true"
```

Classe di archiviazione per iSCSI

Utilizza questo esempio per configurare StorageClass per volumi tramite iSCSI:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  provisioningType: "thin"
  snapshots: "true"
```

Classe di archiviazione che utilizza NFSv3 e AWS Bottlerocket

Per eseguire il provisioning di volumi NFSv3 su AWS Bottlerocket, aggiungere i necessari `mountOptions` alla classe storage:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
mountOptions:
  - nfsvers=3
  - nolock
```

Attributi di Trident StorageClass

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

Attributo	Tipo	Valori	Offerta	Richiesta	Supportato da
BackendType	stringa	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	Il pool appartiene a questo tipo di backend	Backend specificato	Tutti i driver
snapshot	bool	vero, falso	Il pool supporta volumi con snapshot	Volume con snapshot attivate	ontap-nas, ontap-san, solidfire-san
cloni	bool	vero, falso	Il pool supporta la clonazione dei volumi	Volume con cloni attivati	ontap-nas, ontap-san, solidfire-san
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 da ONTAP Select o FSx for ONTAP

Per ulteriori informazioni sull'interazione delle classi di archiviazione con i `PersistentVolumeClaim` parametri e per il controllo del provisioning dei volumi da parte di Trident, fare riferimento "Kubernetes e Trident Objects"a.

Crea la classe di archiviazione

Una volta configurato il StorageClass, puoi crearlo in Kubernetes.

Fasi

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

```
kubectl create -f storage-class-ontapnas.yaml
```

2. Ora dovresti vedere una classe storage **Basic-csi** in Kubernetes e Trident, e Trident dovrebbe aver scoperto i pool nel back-end.

```
kubectl get sc basic-csi
```

```
NAME         PROVISIONER             AGE
```

```
basic-csi    csi.trident.netapp.io   15h
```

Fornire volumi SMB

È possibile eseguire il provisioning dei volumi SMB utilizzando il `ontap-nas` driver. Tuttavia, per farlo è necessario completare questi passaggi: "Preparatevi al provisioning dei volumi SMB".

Configurazione avanzata del backend ed esempi

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
`managementLIF`	Indirizzo IP di un cluster o LIF di gestione SVM È possibile specificare Un nome di dominio completo (FQDN). Può essere impostato in modo da utilizzare gli indirizzi IPv6 se 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]. Se fornisci il `fsxFilesystemID` sotto `aws` il campo, non devi fornire il `managementLIF`, perché Trident recupera le informazioni SVM `managementLIF` da AWS. Pertanto, devi fornire le credenziali a un utente sotto la SVM (ad esempio, vsadmin) e tale utente deve avere un `vsadmin` ruolo.	"10,0.0,1", "[2001:1234:abcd::fefe]"

Parametro	Descrizione	Esempio
`dataLIF`	Indirizzo IP del protocollo LIF. **ONTAP NAS drivers**: NetApp consiglia di specificare dataLIF. Se non fornito, Trident recupera i dataLIF dall'SVM. È possibile specificare un fully-qualified domain name (FQDN) da utilizzare per le operazioni di montaggio NFS, consentendo di creare un round-robin DNS per il bilanciamento del carico su più dataLIF. Può essere modificato dopo l'impostazione iniziale. **ONTAP SAN drivers**: non specificare per iSCSI. Trident utilizza ONTAP Selective LUN Map per individuare i LIF iSCI necessari per stabilire una sessione multipath. Viene generato un avviso se dataLIF è definito esplicitamente. Può essere impostato per utilizzare indirizzi IPv6 se 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].	
`autoExportPolicy`	Abilita la creazione e l'aggiornamento automatici dei criteri di esportazione [booleano]. Utilizzando le `autoExportPolicy` opzioni e `autoExportCIDRs`, Trident può gestire automaticamente i criteri di esportazione.	`false`
`autoExportCIDRs`	Elenco di CIDR per filtrare gli IP dei nodi di Kubernetes rispetto a quando `autoExportPolicy` è attivato. Utilizzando le `autoExportPolicy` opzioni e `autoExportCIDRs`, Trident può gestire automaticamente i criteri 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	""

Parametro	Descrizione	Esempio
`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.
`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.** Fornito `fsxadmin` e `vsadmin` non contiene le autorizzazioni necessarie per recuperare l'utilizzo dell'aggregato e limitarlo mediante Trident.	Non utilizzare.
`limitVolumeSize`	Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore. Limita anche le dimensioni massime dei volumi gestiti per qtree e LUN e l' `qtreesPerFlexvol`opzione consente di personalizzare il numero massimo di qtree per FlexVol volume	"" (non applicato per impostazione predefinita)
`lunsPerFlexvol`	Il numero massimo di LUN per FlexVol volume deve essere compreso nell'intervallo [50, 200]. Solo SAN.	"`100""

Parametro	Descrizione	Esempio
`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
`nfsMountOptions`	Elenco separato da virgole delle opzioni di montaggio NFS. Le opzioni di montaggio per volumi persistenti di Kubernetes vengono normalmente specificate in classi di storage, ma se non sono specificate opzioni di montaggio in una classe di storage, Trident tornerà all'utilizzo delle 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, Trident non imposterà 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 volume, devono essere compresi nell'intervallo [50, 300]	`"200"`
`smbShare`	È possibile specificare uno dei seguenti elementi: Il nome di una condivisione SMB creata utilizzando la console di gestione Microsoft o l'interfaccia CLI di ONTAP oppure un nome per consentire a 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. Quando è impostato su `true`, Trident utilizza le API REST 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 all' `ontap` applicazione. Ciò è soddisfatto dai ruoli predefiniti `vsadmin` e `cluster-admin` .	`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 di FSX SVM da memorizzare in AWS Secrets Manager. - `name`: Amazon Resource Name (ARN) del segreto, che contiene le credenziali di SVM. - `type`: Impostare su `awsarn`. Per ulteriori informazioni, fare riferimento "Creare un segreto AWS Secrets Manager" a.	

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 prenotazione dello spazio; "nessuno" (sottile) o "volume" (spesso)	`none`
`snapshotPolicy`	Policy di Snapshot da utilizzare	`none`

Parametro	Descrizione	Predefinito
`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 criteri QoS con Trident richiede ONTAP 9.8 o versioni successive. È necessario utilizzare un gruppo di criteri QoS non condiviso e garantire che il gruppo di criteri venga applicato singolarmente a ciascun componente. Un gruppo di policy QoS condiviso impone un 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`
`encryption`	Abilitare la crittografia del volume NetApp (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 abilitato sul backend, qualsiasi volume sottoposto a provisioning in Trident sarà abilitato NAE. Per ulteriori informazioni, fare riferimento a: "Come funziona Trident con NVE e NAE".	`false`
`luksEncryption`	Abilita la crittografia LUKS. Consulta "Utilizzo di Linux Unified Key Setup (LUKS)". Solo SAN.	""
`tieringPolicy`	Policy di tiering da utilizzare `none`	
`unixPermissions`	Per i nuovi volumi. **Lasciare vuoto per i volumi SMB.**	""

Parametro	Descrizione	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`.

Configurare un PVC

Questa sezione include istruzioni su come creare un PersistentVolumeClaim (PVC) che utilizza la StorageClass di Kubernetes configurata per richiedere un PV. In caso di successo, è possibile montare il PV su un pod.

Creare il PVC

Un "*PersistentVolumeClaim*" (PVC) è una richiesta di accesso al PersistentVolume sul cluster. Il PVC può essere configurato per richiedere storage di una certa dimensione o modalità di accesso. Utilizzando il StorageClass associato, l'amministratore del cluster può controllare più della sola dimensione e modalità di accesso della PersistentVolume, come ad esempio le prestazioni o il livello di servizio.

Dopo aver creato il backend Trident e la StorageClass, è possibile creare un PVC. Una volta creato il PVC, è possibile montare il volume in un pod.

Manifesti campione

I seguenti esempi mostrano le opzioni di configurazione di base del PVC.

PVC con accesso RWX

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

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-gold
```

PVC utilizzando l'esempio iSCSI

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

```
kind: PersistentVolumeClaim
```

```
apiVersion: v1
metadata:
  name: pvc-san
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: protection-gold
```

Crea PVC

Fasi

1. Creare il PVC.

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

2. Verificare lo stato del PVC.

```
kubectl get pvc
```

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

Per ulteriori informazioni sull'interazione delle classi di archiviazione con i `PersistentVolumeClaim` parametri e per il controllo del provisioning dei volumi da parte di Trident, fare riferimento "Kubernetes e Trident Objects"a.

Distribuisci un'applicazione

Una volta creata la classe di archiviazione e il PVC, è possibile montare il PV su un pod. Questa sezione elenca il comando e la configurazione di esempio per collegare il PV a un pod.

Distribuisci un'applicazione di esempio

Fasi

1. Montare il volume in un pod.

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

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
```

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

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

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

```
Filesystem                                                      Size
Used  Avail Use%  Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06  1.1G
320K  1.0G   1%   /my/mount/path
```

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

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

Configurare il componente aggiuntivo Trident EKS su un cluster EKS

NetApp 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 NetApp 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.

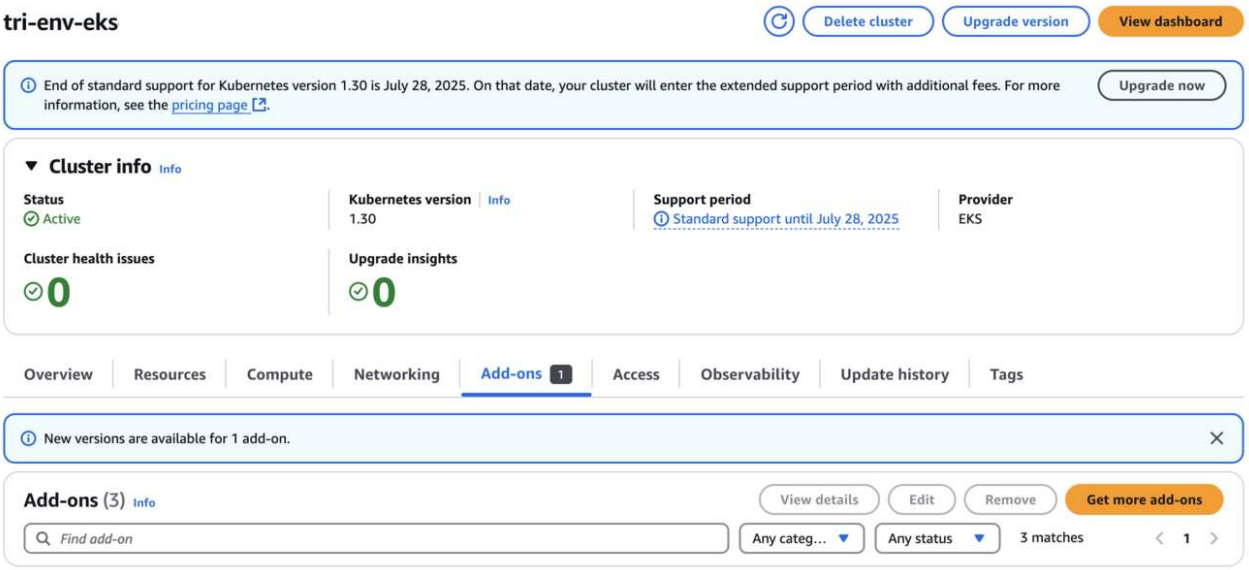
Prerequisiti

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

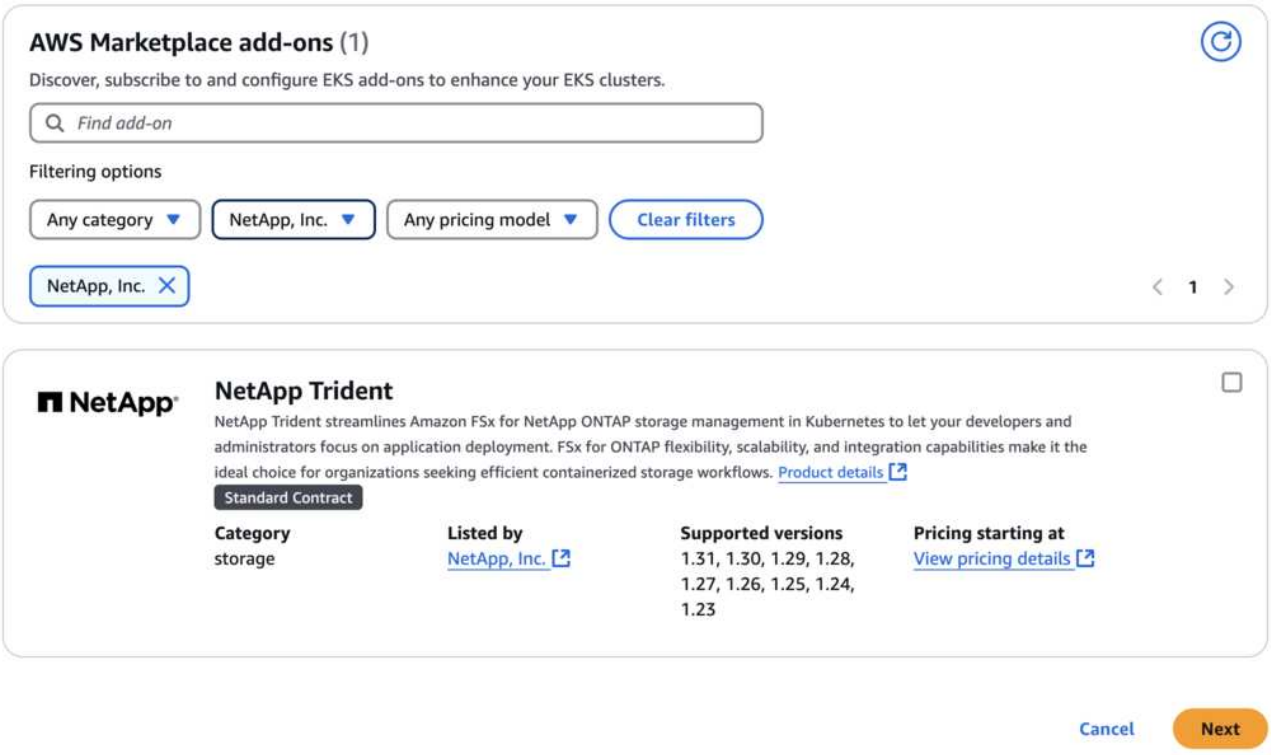
- Un account cluster Amazon EKS con autorizzazioni per l'uso dei componenti aggiuntivi. Fare riferimento alla "Componenti aggiuntivi Amazon EKS".
- Autorizzazioni AWS nel marketplace AWS:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Tipo 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

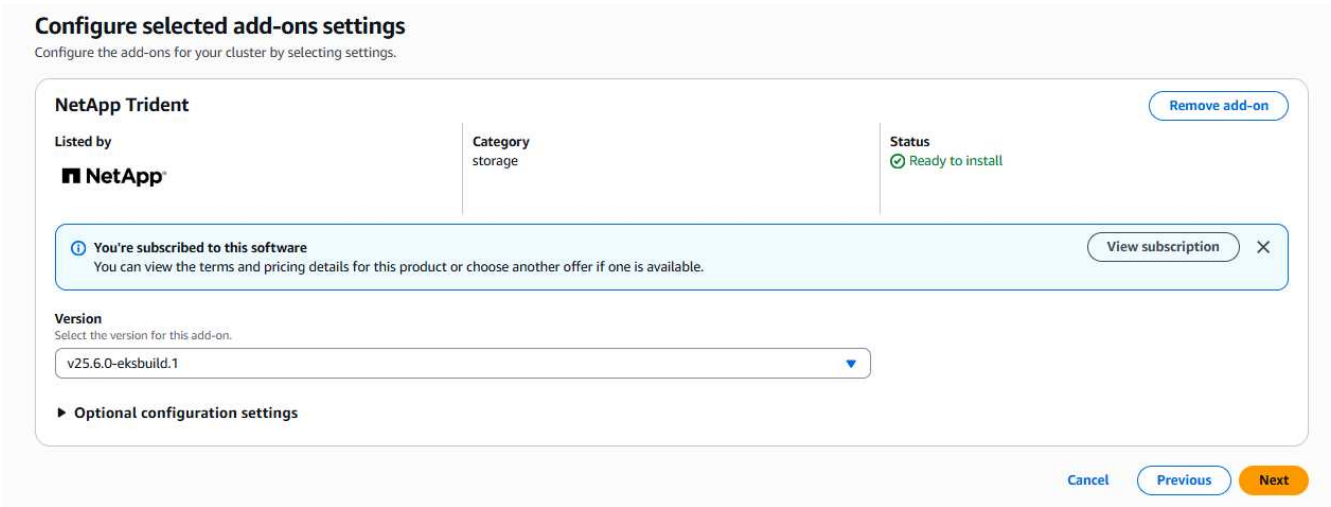
1. Assicurati di creare il ruolo IAM e il segreto AWS per abilitare i pod EKS per accedere alle risorse AWS. Per istruzioni, vedere "Creare un ruolo IAM e un segreto AWS".
2. Sul tuo cluster EKS Kubernetes, accedi alla scheda **Add-on**.



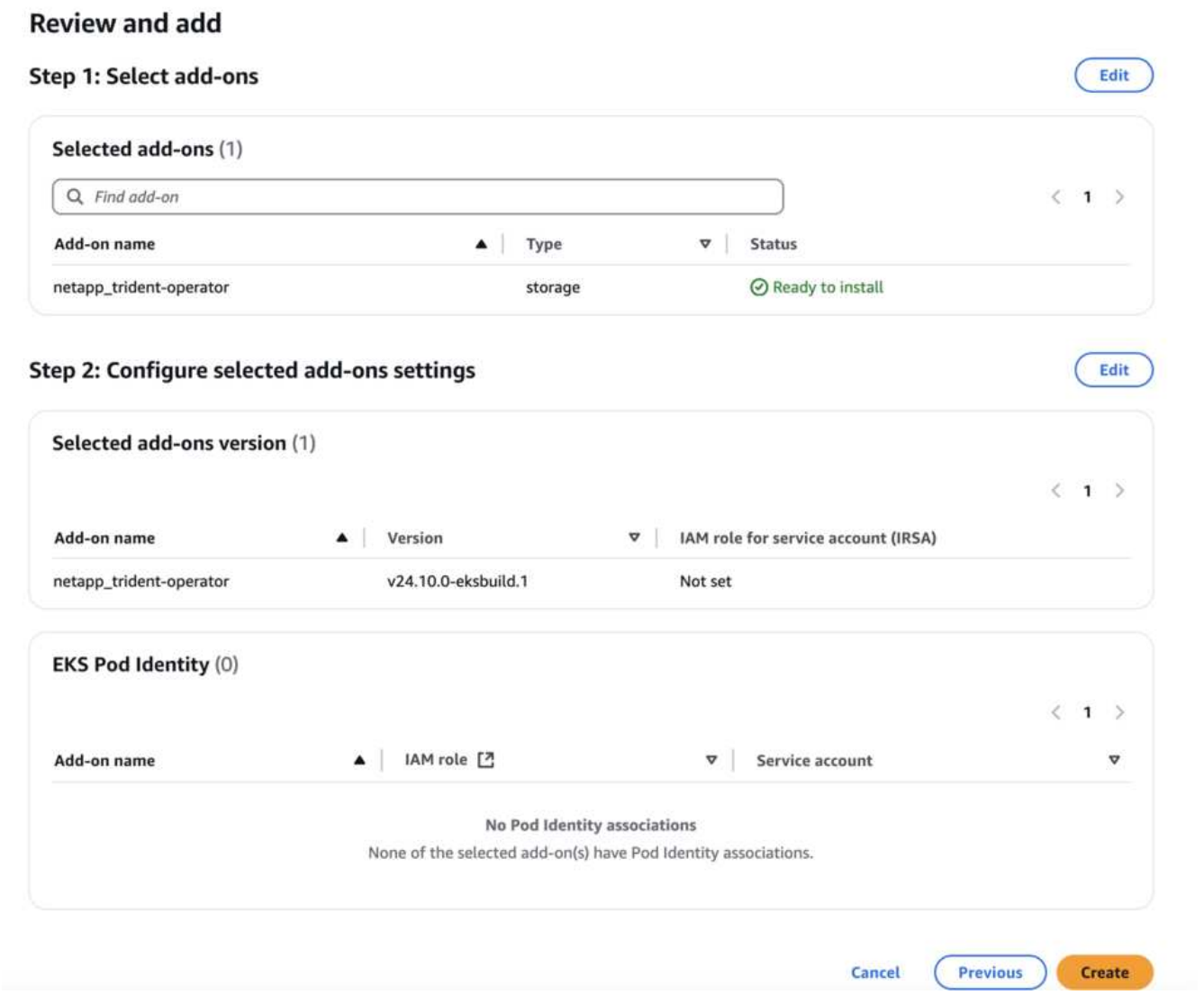
3. Vai su **componenti aggiuntivi di AWS Marketplace** e scegli la categoria *storage*.



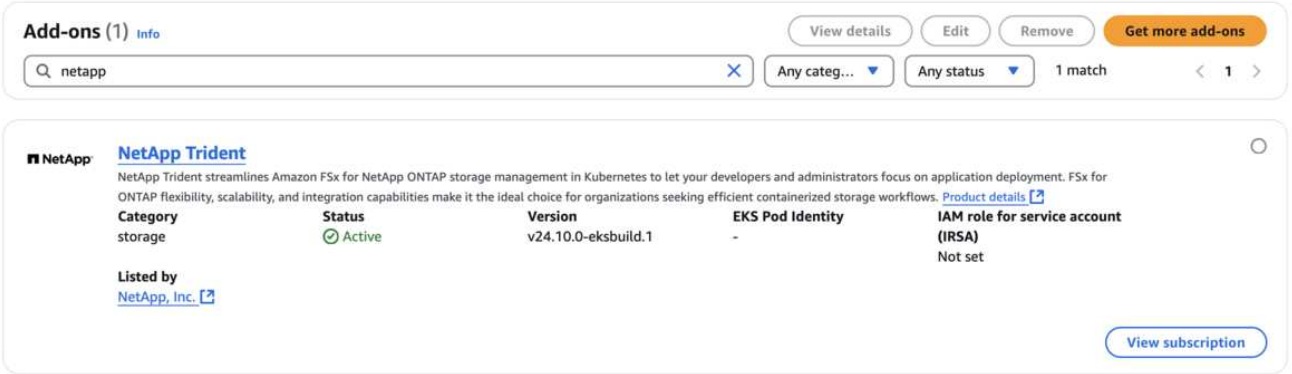
4. Individuare **NetApp Trident** e selezionare la casella di controllo del componente aggiuntivo Trident, quindi fare clic su **Avanti**.
5. Scegliere la versione desiderata del componente aggiuntivo.



6. Configurare le impostazioni aggiuntive richieste.



7. Se si utilizza IRSA (ruoli IAM per l'account di servizio), fare riferimento ai passaggi di configurazione aggiuntivi"qui" .
8. Selezionare **Crea**.
9. Verificare che lo stato del componente aggiuntivo sia *attivo*.



10. Eseguire il seguente comando per verificare che Trident sia installato correttamente nel cluster:

```
kubectl get pods -n trident
```

11. Continua la configurazione e configura il backend di storage. Per informazioni, vedi "Configurare il backend di archiviazione".

Installare/disinstallare il componente aggiuntivo Trident EKS utilizzando la CLI

Installare il componente aggiuntivo NetApp Trident EKS utilizzando la CLI:

Il seguente comando di esempio installa il componente aggiuntivo Trident EKS: `eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.0-eksbuild.1` (con una versione dedicata)

Il seguente comando di esempio installa il componente aggiuntivo Trident EKS versione 25.6.1: `eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.1-eksbuild.1` (con una versione dedicata)

Il seguente comando di esempio installa il componente aggiuntivo Trident EKS versione 25.6.2: `eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.2-eksbuild.1` (con una versione dedicata)

Disinstallare il componente aggiuntivo NetApp Trident EKS utilizzando CLI:

Il seguente comando disinstalla il componente aggiuntivo Trident EKS:

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

Crea backend con kubectl

Un backend definisce la relazione tra Trident e un sistema di storage. Spiega a Trident come comunicare con quel sistema storage e come Trident dovrebbe eseguire il provisioning dei volumi da esso. Dopo l'installazione di Trident, il passaggio successivo consiste nella creazione di un backend. La `TridentBackendConfig` definizione risorsa personalizzata (CRD) ti consente di creare e gestire i backend Trident direttamente attraverso l'interfaccia di Kubernetes. Puoi farlo utilizzando `kubectl` o l'equivalente strumento CLI per la tua distribuzione Kubernetes.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, , `tbconfig` `tbackendconfig`) È un CRD in primo piano, con nome, che consente di gestire backend Trident utilizzando `kubectl`. Gli amministratori di Kubernetes e dello storage possono ora creare e gestire i backend direttamente attraverso l'interfaccia a riga di comando di Kubernetes senza richiedere un'utility a riga di comando dedicata (`tridentctl`).

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

- Trident crea automaticamente un backend in base alla configurazione fornita. Questo è rappresentato internamente come a `TridentBackend` (`tbe`, `tridentbackend`) CR.
- Il `TridentBackendConfig` è associato in modo univoco a un `TridentBackend` creato da 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.

ATTENZIONE	`TridentBackend` I CRS vengono creati automaticamente da Trident. Non è possibile modificarle. Se si desidera aggiornare i backend, modificare l' `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.

NOTA	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`.
SUGGERIMENTO	I backend creati con `tridentctl` non hanno un oggetto associato `TridentBackendConfig`. È possibile scegliere di gestire tali backend con `kubectl` creando una `TridentBackendConfig` CR. Occorre prestare attenzione a specificare parametri di configurazione identici (come `spec.backendName`, , `spec.storagePrefix`, `spec.storageDriverName` e così via). Trident associa automaticamente il nuovo creato `TridentBackendConfig` al backend preesistente.

Panoramica dei passaggi

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

1. Crea un "Kubernetes Secret". il segreto contiene le credenziali che Trident deve avere per comunicare con il cluster/servizio di archiviazione.
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
Elemento (NetApp HCI/SolidFire)	Endpoint	MVIP per il cluster SolidFire con credenziali tenant
ONTAP	nome utente	Nome utente per la connessione al cluster/SVM. Utilizzato per l'autenticazione basata su credenziali
ONTAP	password	Password per la connessione al cluster/SVM. Utilizzato per l'autenticazione basata su credenziali
ONTAP	ClientPrivateKey	Valore codificato in base64 della chiave privata del client. Utilizzato per l'autenticazione basata su certificato
ONTAP	ChapNomeUtente	Nome utente inbound. Obbligatorio se useCHAP=true. Per `ontap-san` e. `ontap-san-economy`
ONTAP	ChapInitialatorSecret	Segreto iniziatore CHAP. Obbligatorio se useCHAP=true. Per `ontap-san` e. `ontap-san-economy`
ONTAP	ChapTargetNomeUtente	Nome utente di destinazione. Obbligatorio se useCHAP=true. Per `ontap-san` e. `ontap-san-economy`
ONTAP	ChapTargetInitialatorSecret	CHAP target Initiator secret. Obbligatorio se useCHAP=true. Per `ontap-san` e. `ontap-san-economy`

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

Fase 2: Creare `TridentBackendConfig` CR

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

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

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
```

```
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Fase 3: Verificare lo stato di `TridentBackendConfig` CR

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

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

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

La fase può assumere uno dei seguenti valori:

- `Bound`: Il `TridentBackendConfig` CR è associato a un backend e contiene tale backend `configRef` impostare su `TridentBackendConfig` 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 sul backend non sono presenti PVC (Persistent Volume Request), l'eliminazione di `TridentBackendConfig` comporterà l'eliminazione del back-end e della CR da parte di Trident `TridentBackendConfig`.
  - 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`: Trident non è in grado di determinare lo stato o l'esistenza del backend associato al

`TridentBackendConfig` CR. Ad esempio, se il server API non risponde o se manca il `tridentbackends.trident.netapp.io` CRD. 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 il `backendName` e il `backendUUID` del backend creato in risposta al `TridentBackendConfig` CR. Il `lastOperationStatus` campo rappresenta lo stato dell'ultima operazione del `TridentBackendConfig` CR, che può essere attivata dall'utente (ad esempio, un elemento modificato dall'utente in) o attivata da Trident (ad esempio, `spec` durante il riavvio di Trident). Può essere riuscito o non riuscito. `phase` Rappresenta lo stato della relazione tra `TridentBackendConfig` CR e backend. Nell'esempio precedente, `phase` ha il valore associato, il che significa che la `TridentBackendConfig` CR è associata al backend.

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

ATTENZIONE	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

Gestisci i backend di Trident con kubectl

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

Eliminare un backend

Eliminando un `TridentBackendConfig`, si ordina a Trident di eliminare/conservare i backend (in base a `deletionPolicy`). Per eliminare un backend, assicurarsi che `deletionPolicy` sia impostato su Elimina. Per eliminare solo il `TridentBackendConfig`, assicurarsi che `deletionPolicy` sia impostato su Mantieni. In questo modo si garantisce che il backend sia ancora presente e che possa essere gestito utilizzando `tridentctl`.

Eseguire il seguente comando:

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

Trident non elimina i segreti di Kubernetes 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, è necessario aggiornare il segreto Kubernetes utilizzato nell' `TridentBackendConfig`oggetto. 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
    ```

NOTA

- 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.

Gestisci i backend di Trident 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 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
   ```

NOTA	Se Trident ha eseguito il provisioning di volumi e Snapshot da questo backend che ancora esistono, l'eliminazione del backend impedisce il provisioning di nuovi volumi da parte dell'IT. Il backend continuerà ad esistere in uno stato di "eliminazione".

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`.

Scegli un metodo di gestione del backend Trident

Scopri i diversi modi di gestire i backend in 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 Trident che pianifica i volumi e li utilizza. 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 di cui Trident ha bisogno per comunicare con il cluster/servizio di archiviazione.
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 deve essere impostato su `retain`. In questo modo, quando si elimina un `TridentBackendConfig` CR, la definizione di backend sarà ancora presente e potrà 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
```

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

All'eliminazione dell' `TridentBackendConfig`oggetto, Trident lo rimuove semplicemente senza eliminare effettivamente il backend stesso.

Creare e gestire classi di archiviazione

Crea una classe di storage Trident

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

Configurare un oggetto Kubernetes StorageClass

```
https://kubernetes.io/docs/concepts/storage/storage-classes/["Oggetto
Kubernetes StorageClass"^]Identifica Trident come provisioner utilizzato
per quella classe e istruisce Trident su come effettuare il provisioning
di un volume. Ad esempio:
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
mountOptions:
  - nfsvers=3
  - nolock
parameters:
  backendType: "ontap-nas"
  media: "ssd"
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Per ulteriori informazioni sull'interazione delle classi di archiviazione con i `PersistentVolumeClaim` parametri e per il controllo del provisioning dei volumi da parte di Trident, fare riferimento "Kubernetes e Trident Objects"a.

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 dovresti vedere una classe storage **Basic-csi** in Kubernetes e Trident, e Trident dovrebbe aver scoperto i pool nel back-end.

```
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

Trident fornisce "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
```

Gestisci le classi di storage di Trident

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 archiviazione sincronizzata di Trident, eseguire il comando seguente:

```
tridentctl get storageclass
```

- Per visualizzare i dettagli della classe di archiviazione sincronizzata di Trident, eseguire il comando seguente:

```
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.

NOTA	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 è possibile rispondere con il JSON che `tridentctl` emette per gli oggetti back-end Trident. In questo modo viene utilizzata l' `jq`utilità, 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 tramite questa classe di storage non verranno toccati e Trident continuerà a gestirli.

NOTA Trident applica uno spazio vuoto `fsType` ai volumi che crea. Per i backend iSCSI, si consiglia di applicare `parameters.fsType` in StorageClass. È necessario eliminare gli StorageClasses esistenti e ricrearli con `parameters.fsType` specificato.

Provisioning e gestione dei volumi

Esegui il provisioning di un volume Trident

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

Panoramica

Un "*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 il PVC, è possibile montare il volume in un pod.

Creare il PVC

Fasi

1. Creare il PVC.

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

2. Verificare lo stato del PVC.

```
kubectl get pvc
```

```
NAME          STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS AGE
pvc-storage   Bound    pv-name    1Gi        RWO                    5m
```

1. Montare il volume in un pod.

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

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

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

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

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

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

Manifesti campione

▼ **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: storage
      persistentVolumeClaim:
       claimName: pvc-storage
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: storage
```

Configurazione NVMe/TCP di base

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

Per ulteriori informazioni sull'interazione delle classi di archiviazione con i `PersistentVolumeClaim` parametri e per il controllo del provisioning dei volumi da parte di Trident, fare riferimento "Kubernetes e Trident Objects"a.

Sposta un volume

A partire da Trident 26.06, un amministratore dello storage può spostare in modo non distruttivo un volume iSCSI su un nodo ONTAP e un aggregato diversi all'interno dello stesso cluster ONTAP mentre il carico di lavoro che lo utilizza continua a funzionare. Richiedi lo spostamento in modo dichiarativo con una `TridentVolumeMove` custom resource o l' `tridentctl` equivalente, e Trident coordina ONTAP e l'host così che l'I/O non venga mai interrotto.

Requisiti

La funzione di spostamento del volume si applica al driver `ontap-san` per i volumi iSCSI solo. I protocolli NVMe e NAS non sono supportati.

Prima di spostare un volume, assicurati che siano soddisfatti i seguenti requisiti:

- Il backend utilizza il `ontap-san` driver con l'API REST ONTAP.
- L'ONTAP cluster ha almeno due nodi.
- Hai uno dei seguenti set di credenziali:
  - ONTAP 9.19.1 o versioni successive se usi il ruolo ONTAP personalizzato dedicato.
  - Credenziali di amministratore del cluster nelle versioni precedenti di ONTAP.

Come funziona

Trident sposta il volume (LUN) tra due posizioni all'interno dello stesso ONTAP cluster. Prima che i dati vengano spostati, Trident aggiunge nuovi percorsi iSCSI alla destinazione, affiancandoli a quelli esistenti, così il dispositivo multipath dell'host non perde mai l'accesso. Dopo che ONTAP ha completato lo spostamento, Trident elimina i percorsi di origine obsoleti.

Trident gestisce lo spostamento attraverso una serie ordinata di stati visibili nella `TridentVolumeMove` risorsa, così puoi seguire l'avanzamento dello spostamento.

Sposta un volume

Identifica il volume e le posizioni

1. Individua il nome del PV da spostare. Il nome del PV è anche il nome della risorsa di spostamento.
2. Dal cluster ONTAP, annota il nodo e l'aggregato correnti (di origine) per quel volume e scegli il nodo e l'aggregato di destinazione.

Inizia lo spostamento

Puoi usare entrambi i metodi. Entrambi creano la stessa `TridentVolumeMove`risorsa, il cui nome è il nome del PV.

Utilizzo di tridentctl

```
tridentctl create volume-move \
  --volume      pvc-8a814d62-bd58-4253-b0d1-82f2885db671 \
```

```
  --source-node ontap-node-01 \
  --source-pool aggr1_node01 \
  --target-node ontap-node-02 \
  --target-pool aggr1_node02 \
  -n trident
```

`tvm` è un alias per `volume-move`. Aggiungi `--delete-after-success 10m` per rimuovere automaticamente la risorsa dopo che lo spostamento ha avuto successo.

Utilizzando una risorsa TridentVolumeMove

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeMove
metadata:
  name: pvc-8a814d62-bd58-4253-b0d1-82f2885db671   # the PV to move
  namespace: trident
spec:
  sourceNode: ontap-node-01
  sourcePool: aggr1_node01
  targetNode: ontap-node-02
  targetPool: aggr1_node02
  # deleteAfterSuccess: 10m   # optional
```

Applica la risorsa:

```
kubectl apply -f tridentvolumemove.yaml
```

Campi di specifica

`sourceNode`/ `targetNode`

Nomi dei nodi ONTAP (non UUID).

`sourcePool`/ `targetPool`

Nomi degli aggregate ONTAP.

`deleteAfterSuccess`

Durata facoltativa, ad esempio `30s` o `10m`. Imposta su `0` per eliminare subito la risorsa, oppure ometti per conservarla.

Monitora lo spostamento

Monitora i progressi utilizzando il `tvm`nome breve o `tridentvolumemove`:

```
kubectl get tridentvolumemove -n trident
```

Per visualizzare lo stato attuale e i progressi per ciascun nodo, descrivi la risorsa:

```
kubectl describe tridentvolumemove <pv-name> -n trident
```

Il volume può avere collegamenti simultanei durante lo `Moving` stato, perché ONTAP può impiegare del tempo per completare lo spostamento di un volume di grandi dimensioni. Per garantire uno spostamento senza interruzioni, Trident coordina tra il Trident controller e il Trident node. Ogni stato esegue i seguenti passaggi in ordine, terminando con `Succeeded` o `Failed`.

Stato	Che cosa sta succedendo
`Pending`	Trident accetta e convalida la richiesta di spostamento e raccoglie le informazioni di cui ha bisogno prima che vengano apportate modifiche allo storage.
`ControllerStaging`	Trident prepara la posizione di destinazione. Mappa le interfacce di archiviazione del nodo di destinazione sulla LUN e aggiorna Kubernetes in modo che un futuro riavvio del nodo si riconnetta alla posizione corretta.
`NodeStaging`	Su ciascun host in cui è montato il volume, Trident aggiunge i nuovi percorsi di archiviazione dal nodo di destinazione accanto ai percorsi esistenti, in modo che l'I/O del pod continui su entrambi i set di percorsi.
`Moving`	ONTAP copia e trasferisce i dati del volume all'aggregato di destinazione. Questa è la fase più lunga e ONTAP la gestisce interamente, mentre Trident esegue il polling fino al completamento. Durante questa fase sono consentite nuove pubblicazioni.
`ControllerUnstaging`	Il trasferimento dei dati è completato. Trident aggiorna Kubernetes per rimuovere i percorsi di origine dallo stato desiderato e poi scollega le interfacce di storage del nodo di origine dalla LUN.
`NodeUnstaging`	Su ciascun host, Trident rimuove i percorsi sorgente obsoleti. All'host rimangono solo i percorsi di destinazione e il dispositivo multipath continua a gestire le operazioni di I/O senza interruzioni.
`Succeeded`	Il volume è completamente spostato e online sulla destinazione. Trident conserva la `TridentVolumeMove` risorsa oppure la elimina automaticamente se imposti `deleteAfterSuccess`.
`Failed`	Si è verificato un errore irreversibile che ha interrotto lo spostamento o ne ha impedito l'avvio. Trident ripristina lo stato del volume per consentire la ripresa delle normali operazioni e mantiene la risorsa con un messaggio di errore. Esamina il messaggio, risolvi la causa principale, elimina la risorsa e riprova.

L' `.status.attachments`elenco riporta lo stato di avanzamento di ciascun nodo collegato nel suo `state`campo:

Stato di collegamento	Significato
`Pending`	Nodo identificato come collegato; il lavoro non è ancora iniziato.
`Bridged`	Nuovi percorsi di destinazione aggiunti accanto a quelli esistenti.
`Migrated`	Il traffico ora scorre sui nuovi percorsi.

Stato di collegamento	Significato
`Cleaned`	Percorsi sorgente obsoleti rimossi dal nodo.
`Detached`	Il nodo si è staccato durante lo spostamento ed è stato escluso da ulteriori valutazioni mentre Trident verifica se l'aggancio è pronto per procedere al passo successivo.
`Failed`	Errore del terminale su questo nodo. Guarda il messaggio.

Pulisci la risorsa di spostamento

Se imposti `deleteAfterSuccess`, Trident rimuove automaticamente la risorsa. Altrimenti, puoi eliminarla manualmente. Prima di richiedere un altro spostamento sullo stesso volume, assicurati che non esista alcuna risorsa `TridentVolumeMove` per quel volume né in uno stato `Succeeded` né in uno stato `Failed`.

NOTA Trident non elimina mai automaticamente le operazioni non riuscite. Trident conserva una `Failed` risorsa come record di controllo. Dopo che hai risolto la causa principale, elimina manualmente la risorsa prima di riprovare.

Espandi i volumi Trident

Trident offre agli utenti di Kubernetes la possibilità di espandere i propri volumi dopo averli creati. Trova informazioni sulle configurazioni necessarie per espandere i volumi iSCSI, NFS, SMB, NVMe/TCP e FC.

Espandere un volume iSCSI

È possibile espandere un volume persistente iSCSI (PV) utilizzando il provisioning CSI.

NOTA 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
```

Trident crea un volume persistente (PV) e lo associa a questa dichiarazione di volume persistente (PVC).

```
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, Trident espande il volume sul backend dello storage, esegue una nuova scansione del dispositivo e ridimensiona il file system.
- Quando si tenta di ridimensionare un PV non collegato, 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 convalidare il corretto funzionamento dell'espansione controllando le dimensioni del PVC, del PV e del volume 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 FC

È possibile espandere un volume persistente FC (PV) utilizzando il provisioner CSI.

NOTA	L'espansione del volume FC è supportata dal `ontap-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
```

Trident crea un volume persistente (PV) e lo associa a questa dichiarazione di volume persistente (PVC).

```
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. Quando si ridimensiona un FV FC, esistono due scenari:

- Se il PV è collegato a un pod, Trident espande il volume sul backend dello storage, esegue una nuova scansione del dispositivo e ridimensiona il file system.
- Quando si tenta di ridimensionare un PV non collegato, 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 convalidare il corretto funzionamento dell'espansione controllando le dimensioni del PVC, del PV e del volume 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

Trident supporta l'espansione del volume per i PV NFS forniti su `ontap-nas` , `ontap-nas-economy` , `ontap-nas-flexgroup` , E `azure-netapp-files` backend.

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
```

```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
```

Trident dovrebbe creare un PV NFS da 20 MiB 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 da 20 MiB appena creato a 1 GiB, modificare il PVC e impostare `spec.resources.requests.storage` a 1 GiB:

```
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 convalidare il ridimensionamento corretto controllando le dimensioni del PVC, PV e del volume 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    |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Comprendere i limiti del sottosistema RWX NVMe

ReadWriteMany (RWX) volumi che utilizzano il protocollo NVMe hanno un limite di scalabilità di 64 nodi per volume. Di seguito sono illustrate le limitazioni, viene spiegata l'architettura del sottosistema NVMe coinvolto e vengono delineati i passaggi di risoluzione richiesti.

Comprendere il limite di 64 nodi

Se si prevede di utilizzare volumi ReadWriteMany (RWX) con il protocollo NVMe, un singolo volume RWX NVMe non può essere montato da più di 64 nodi in un cluster Kubernetes.

Non pianificare carichi di lavoro che montano lo stesso RWX NVMe PersistentVolumeClaim su più di 64 nodi.

Questa limitazione si applica solo ai volumi RWX che utilizzano il protocollo NVMe.

Comprendere i modelli di sottosistema NVMe

Modello di sottosistema per volume (versioni di Trident precedenti alla 26.02)

Nelle versioni di Trident precedenti alla 26.02, i volumi RWX NVMe vengono forniti utilizzando un modello di sottosistema per volume. Ogni volume RWX NVMe è mappato al proprio sottosistema NVMe dedicato su ONTAP.

Questo modello è semplice, ma presenta un limite di scalabilità inferiore. Nei cluster Kubernetes di grandi dimensioni, i limiti del controller del sottosistema vengono raggiunti rapidamente perché ogni volume RWX consuma un sottosistema dedicato.

Modello super-sottosistema (introdotto in Trident 26.02)

A partire da Trident 26.02, i volumi RWX NVMe utilizzano un modello di super-sottosistema condiviso. Più volumi RWX NVMe condividono lo stesso sottosistema NVMe.

Ogni super-sottosistema supporta fino a 1024 namespace (volumi). Questo modello migliora significativamente la scalabilità per i carichi di lavoro RWX e riduce la probabilità di raggiungere i limiti del sottosistema ONTAP.

Ogni volume RWX NVMe supporta fino a 64 nodi.

Identificare i sintomi di errore

Se crei o colleghi volumi RWX NVMe su larga scala, potresti osservare errori simili ai seguenti:

```
Maximum number of controllers reached. No more controllers can be created.
```

Questo errore indica che è stato raggiunto il limite del controller del sottosistema ONTAP NVMe.

Risolvere gli errori di limite del sottosistema

Per superare le limitazioni del sottosistema per volume e sfruttare il modello super-sottosistema, esegui l'aggiornamento a Trident 26.02 o versioni successive.

Aggiorna Trident per applicare il modello super-subsystem

Per applicare il modello super-sottosistema ai volumi RWX NVMe:

1. Aggiorna Trident alla versione 26.02 o successiva.
2. Riduci a zero le repliche di tutti i pod che utilizzano volumi RWX NVMe.
3. Verificare che nessun carico di lavoro stia utilizzando attivamente volumi RWX NVMe.
4. Aumenta nuovamente il numero di pod.

Questa sequenza di riavvio garantisce che i volumi RWX NVMe siano collegati utilizzando il modello super-subsystem.

- Questa limitazione si applica solo ai volumi RWX che utilizzano il protocollo NVMe.
- Il limite di 64 nodi si applica per ogni volume RWX NVMe.
- Altre modalità di accesso e altri protocolli non sono interessati.

Scalabilità del controller

Trident migliora la scalabilità del controller grazie a una maggiore concorrenza tra più driver di storage. Quando abiliti la scalabilità del controller, il controller Trident elabora le operazioni di storage in parallelo invece di serializzarle, il che aumenta il throughput negli ambienti Kubernetes con molte operazioni simultanee.

Prima di distribuire, determina quali driver Trident supportano la scalabilità del controller nella disponibilità generale e quali driver sono disponibili come anteprima tecnica in Trident 26.10. Questo ti aiuta a prendere decisioni di distribuzione informate e a gestire il rischio. La scalabilità del controller è disabilitata per impostazione predefinita.

Concetti chiave e definizioni

Scalabilità del controller

La scalabilità del controller si riferisce alla capacità del controller Trident di elaborare più operazioni di storage in parallelo, anziché serializzarle dietro un singolo blocco. Queste operazioni includono la creazione, l'eliminazione e il ridimensionamento dei volumi; la creazione e l'eliminazione di snapshot; la pubblicazione e l'annullamento della pubblicazione dei volumi; e la gestione del backend.

Quando abiliti la scalabilità del controller, le operazioni su volumi e backend diversi procedono contemporaneamente. Questo aumenta il throughput e riduce il tempo di esecuzione end-to-end in ambienti con un elevato numero di operazioni PersistentVolumeClaim e VolumeSnapshot simultanee.

Comportamento predefinito (modalità seriale)

Per impostazione predefinita, il controller Trident elabora le operazioni una alla volta. Ogni richiesta di creazione, eliminazione, ridimensionamento o snapshot viene completata prima che Trident avvii la successiva. La modalità seriale è quella predefinita supportata per tutte le installazioni e non richiede configurazione.

La modalità seriale è sufficiente per la maggior parte dei carichi di lavoro. Abilita la scalabilità del controller solo quando il volume delle operazioni crea un accumulo durante l'elaborazione in modalità seriale.

Supporto per la scalabilità del controller

Trident supporta la scalabilità del controller a diversi livelli di maturità per diversi driver di storage.

Disponibilità generale

I seguenti driver supportano la scalabilità del controller alla disponibilità generale in Trident 26.10:

- `ontap-san`
- `ontap-nas`
- `ontap-nas-economy`
- `ontap-nas-flexgroup`
- `ontap-san-economy`
- `google-cloud-netapp-volumes`
- `azure-netapp-files`
- `solidfire-san`

NOTA I `google-cloud-netapp-volumes` e `google-cloud-netapp-volumes-san` driver sono diversi. Trident supporta solo `google-cloud-netapp-volumes`. Non usare `google-cloud-netapp-volumes-san` nelle configurazioni o negli esempi di backend.

Anteprima tecnica

Il seguente driver supporta la scalabilità del controller come funzionalità in anteprima tecnica in Trident 26.10:

- `asa-r2` (SAN e NVMe)

Questo driver presenta le seguenti limitazioni:

- La funzionalità di concorrenza del controller è disponibile solo per valutazione e test.
- Il comportamento può cambiare da una versione all'altra.
- NetApp non consiglia l'utilizzo in ambienti di produzione.

Come funziona enableConcurrency

Quando imposti `enableConcurrency` su `true`, Trident applica l'elaborazione simultanea a tutti i backend gestiti dal controller. L'impostazione si applica a tutti i backend contemporaneamente. Non puoi abilitarla per singoli backend o singoli driver.

Ogni backend configurato deve utilizzare un driver presente nell'elenco di disponibilità generale o di anteprima tecnica. Se un backend utilizza un driver non supportato, Trident non si avvia e Trident non aggiunge un backend che utilizza un driver non supportato.

Per ripristinare l'avvio, rimuovi o riconfigura qualsiasi backend che utilizza un driver non supportato oppure disabilita la scalabilità del controller.

Prima di abilitare

IMPORTANTE	Prima di abilitare la scalabilità del controller, assicurati che ogni backend configurato utilizzi un driver presente nell'elenco di disponibilità generale o di anteprima tecnica. Se un backend utilizza un driver non supportato, Trident non si avvia dopo che hai abilitato la funzionalità.

Utilizza la tabella seguente per decidere se abilitare la scalabilità del controller.

Se il tuo ambiente	Poi
Utilizzi solo driver supportati e riscontri un accumulo di operazioni del controller	Abilita la scalabilità del controller.
Utilizza un driver non supportato	Mantieni la modalità seriale predefinita. Non abilitare la scalabilità del controller.
Gestisce bassi volumi di operazioni senza arretrati	Mantieni la modalità seriale predefinita.

Abilita la scalabilità del controller

L' `enableConcurrency`opzione di configurazione controlla la scalabilità del controller. Devi abilitare esplicitamente questa opzione durante l'installazione di Trident o quando aggiorni una distribuzione esistente.

Distribuzione dell'operatore Trident

Per abilitare la scalabilità del controller con l'operatore Trident, imposta `enableConcurrency` su `true` nella `TridentOrchestrator` risorsa personalizzata (CR).

Nuova installazione

Crea o modifica il `TridentOrchestrator` CR e imposta `enableConcurrency` su `true`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
```

```
  enableConcurrency: true
```

Applicare la CR:

```
kubectl apply -f tridentorchestrator_cr.yaml
```

Installazione esistente

Applicare una patch al `TridentOrchestrator` CR esistente per abilitare la scalabilità del controller:

```
kubectl patch torc trident --type=merge -p
'{"spec":{"enableConcurrency":true}}'
```

Verifica che Trident abbia applicato l'impostazione:

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableConcurrency}'
```

Distribuzione Helm

Per abilitare la scalabilità del controller con Helm, impostare il `enableConcurrency` valore su `true`.

Nuova installazione

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace --set enableConcurrency=true
```

Installazione esistente

```
helm upgrade trident netapp-trident/trident-operator --namespace trident
--set enableConcurrency=true
```

In alternativa, imposta `enableConcurrency` su `true` in un file `values.yaml` personalizzato:

```
# values.yaml
enableConcurrency: true
```

Quindi installa o aggiorna utilizzando il file dei valori:

```
helm install trident netapp-trident/trident-operator --namespace trident
```

```
--create-namespace -f values.yaml
```

implementazione tridentctl

Per abilitare la scalabilità del controller con `tridentctl`, passare il `--enable-concurrency` flag durante l'installazione.

Nuova installazione

```
tridentctl install -n trident --enable-concurrency
```

Installazione esistente

Per abilitare la scalabilità del controller su una distribuzione `tridentctl`esistente, disinstalla Trident e reinstallalo con il flag:

```
tridentctl uninstall -n trident
tridentctl install -n trident --enable-concurrency
```

Verifica che la scalabilità del controller sia abilitata

Dopo aver abilitato la scalabilità del controller, verifica che il controller Trident sia in esecuzione con la concorrenza abilitata. Controlla i log del pod del controller:

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

L'output include una voce di registro che conferma che la concorrenza è abilitata.

Comportamento della concorrenza

Quando la scalabilità del controller è abilitata, il controller Trident adotta il seguente comportamento:

- Trident sostituisce il blocco globale singolo con un blocco granulare per risorsa.
- Trident serializza le operazioni che modificano la stessa risorsa per mantenere la coerenza dei dati.
- Le operazioni che si limitano a leggere da una risorsa procedono contemporaneamente ad altre operazioni di lettura su quella risorsa.
- Trident limita le richieste API ONTAP simultanee a 20 per LIF di gestione per evitare il sovraccarico dei sistemi di storage di backend.
- Se più backend condividono lo stesso LIF di gestione, condividono questo limite di 20 richieste.

Avvertenze e limitazioni

Le seguenti considerazioni si applicano alla scalabilità del controller in Trident 26.10:

- La scalabilità del controller supporta solo i driver presenti negli elenchi di disponibilità generale e di anteprima tecnica. Per dettagli, vedi Prima di abilitare.

- Il controller Trident gestisce la concorrenza internamente. Questa versione non prevede limiti di concorrenza configurabili dall'utente.
- Il throughput complessivo dipende dal driver di storage in uso, dalla reattività del backend e dalle prestazioni del server API di Kubernetes.
- Un'elevata concorrenza può aumentare il carico sui sistemi di storage back-end.
- Il comportamento di scalabilità del controller non è identico per tutti i driver.
- Il driver di anteprima tecnica può presentare prestazioni incoerenti sotto carico elevato e può cambiare comportamento tra una versione e l'altra.
- Il debug di operazioni concorrenti può risultare più complesso a causa dell'esecuzione parallela. Metriche e log possono mostrare l'output delle operazioni interlacciate.

Raccomandazioni

Prima di abilitare la scalabilità del controller, completa i seguenti passaggi:

1. Conferma che ogni backend configurato utilizza un driver presente nell'elenco di disponibilità generale o di anteprima tecnica.
2. Testa la modifica in un cluster non di produzione prima di applicarla in produzione.
3. Verifica che la scalabilità del controller sia abilitata dopo aver applicato la modifica.

Applica le seguenti raccomandazioni generali quando lavori con la scalabilità del controller:

- Usa i driver di disponibilità generale per gli ambienti di produzione che richiedono un'elevata scalabilità.
- Valuta il driver di anteprima tecnica solo in ambienti non di produzione.
- Monitora le prestazioni del backend e del controller quando operi su larga scala.
- Non dare per scontato l'ordine delle operazioni negli script di automazione.

Espansione automatica del volume

L'espansione automatica del volume consente ai Persistent Volumes forniti da Trident di aumentare automaticamente quando la capacità utilizzata raggiunge una soglia definita. Questa funzionalità riduce il sovraccarico operativo e aiuta a prevenire l'interruzione delle applicazioni causata dall'esaurimento della capacità. L'espansione automatica del volume viene implementata utilizzando Autogrow Policies. Una Autogrow Policy definisce:

- La soglia di utilizzo che attiva l'espansione
- La quantità di cui cresce il volume
- La dimensione massima che il volume può raggiungere

Le policy Trident Autogrow espandono il PVC. Questo differisce da "ONTAP dimensionamento automatico", che espande il FlexVol sottostante senza modificare le dimensioni del PVC.

NOTA	I volumi aumentano automaticamente di dimensione quando viene superata la soglia di utilizzo definita. I volumi non vengono mai ridotti automaticamente.

Requisiti

Prima di configurare l'espansione automatica del volume, assicurarsi che siano soddisfatti i seguenti requisiti:

- Trident 26.02 o versioni successive
- Autorizzazioni di controllo degli accessi basate sui ruoli per creare `TridentAutogrowPolicy` risorse personalizzate
- StorageClasses configurati con `allowVolumeExpansion: true`
- Protocolli ONTAP supportati:
  - Network File System (NFS)
  - Internet Small Computer Systems Interface (iSCSI)
  - Non-Volatile Memory Express (NVMe)
  - Protocollo Fibre Channel (FCP)

Limitazioni

- I volumi raw block ONTAP Non-Volatile Memory Express precedenti a ONTAP 9.16.1 non supportano l'espansione automatica.
- Per i volumi di storage area network, se `growthAmount` è inferiore o uguale a 50 mebibyte, Trident aumenta automaticamente il valore a 51 mebibyte prima del ridimensionamento, a condizione che la dimensione risultante non superi `maxSize`.
- Negli ambienti brownfield, l'espansione automatica potrebbe non funzionare per alcuni volumi esistenti a causa del comportamento di migrazione della pubblicazione del volume.
- Quando un volume raggiunge `maxSize`, non si verifica alcuna ulteriore espansione.
- Protocolli supportati per l'espansione automatica del volume:
  - Network File System (NFS)
  - Internet Small Computer Systems Interface (iSCSI)
  - Non-Volatile Memory Express (NVMe)
  - Protocollo Fibre Channel (FCP)

Provision Volumes con criterio di Autogrow

La policy di Autogrow può essere configurata a due livelli:

- Livello di classe di archiviazione: imposta il valore predefinito per tutti i volumi (usando l'annotazione)
- Livello PVC: sostituisce la classe di archiviazione predefinita (utilizzando annotazione)

Crea una policy di Autogrow

Le Autogrow Policies consentono l'espansione automatica del volume quando i volumi raggiungono una soglia di capacità definita.

Assicurati di avere:

- Trident 26.02 o versioni successive installato
- Autorizzazioni di controllo degli accessi basate sui ruoli per creare `TridentAutogrowPolicy` risorse

- Comprensione dei requisiti di crescita del carico di lavoro

Una Autogrow Policy definisce come i volumi si espandono automaticamente quando raggiungono una soglia di capacità definita.

È possibile creare Autogrow Policies in qualsiasi momento del flusso di lavoro:

- Prima che StorageClasses e i volumi vengano creati
- Dopo che StorageClasses esistono
- Dopo che i volumi sono stati effettuati il provisioning

Questa flessibilità consente di introdurre l'espansione automatica senza ricreare le risorse esistenti.

Specifiche della Autogrow Policy

Le Autogrow Policies sono risorse personalizzate di Kubernetes definite come segue:

Campo	Descrizione	Formato	Obbligatorio	Esempio	Predefinito
nome	Identificatore univoco della policy	Stringa	Sì	production-db-policy	Nessuno
usedThreshold	Percentuale di capacità che innesca l'espansione	Stringa percentuale	Sì	"80%"	Nessuno
growthAmount	Quantità da aumentare quando viene raggiunta la soglia	Percentuale o dimensione	No	"10%" o "5Gi"	"10%"
maxSize	Limite massimo della dimensione del volume	Quantità Kubernetes	No	"500Gi"	Illimitato

Crea una policy di Autogrow

Fasi

1. Crea un file YAML che definisce la tua Autogrow Policy:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

2. Applica la policy al tuo cluster:

```
kubectl apply -f autogrow-policy.yaml
```

3. Verificare che la policy sia stata creata:

```
kubectl get tridentautogrowpolicy standard-autogrow
```

Output previsto

```
NAME                USED THRESHOLD   GROWTH AMOUNT   STATE
standard-autogrow   80%              10%             Success
```

Stati delle policy

Dopo aver creato una policy, Trident convalida la specifica e assegna uno dei seguenti stati:

Stato	Descrizione	Azione richiesta
Successo	La policy è convalidata e pronta per l'uso.	Nessuno.
Non riuscito	Sono stati rilevati errori di convalida.	Rivedi e correggi la specifica.
Eliminazione	L'eliminazione è in corso.	Attendere il completamento.

Associa una policy a un StorageClass

È possibile associare una Autogrow Policy a una StorageClass utilizzando l' `trident.netapp.io/autogrowPolicy`annotazione. Tutti i volumi forniti da quella StorageClass ereditano la policy.

NOTA Il StorageClass deve avere `allowVolumeExpansion: true`.

Fasi

1. Crea o modifica una StorageClass con l'annotazione Autogrow Policy:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

2. Applica il StorageClass:

```
kubectl apply -f storageclass.yaml
```

3. Verifica l'annotazione:

```
kubectl get storageclass ontap-gold -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Output previsto

```
production-db-policy
```

Precedenza delle policy

Quando le annotazioni Autogrow Policy sono impostate sia su una StorageClass che su un PVC, Trident applica le seguenti regole di precedenza:

1. **L'annotazione PVC ha la priorità.** Se un PVC imposta `trident.netapp.io/autogrowPolicy`, quel valore viene sempre utilizzato.
2. **StorageClass annotation si applica solo quando il PVC non ha annotazione.**
3. **Se nessuno dei due ha l'annotazione,** non viene applicata alcuna Autogrow Policy.

Annotazione StorageClass	Annotazione PVC	Comportamento efficace
`trident.netapp.io/autogrowPolicy: standard-agp`	Non impostato	Usa `standard-agp`.
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: logs-policy`	Utilizza `logs-policy` (PVC sostituisce StorageClass).
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: "none"`	No Autogrow Policy (PVC disabilita autogrow).
Non impostato	`trident.netapp.io/autogrowPolicy: dev-policy`	Usi `dev-policy`.
Non impostato	Non impostato	Nessuna Autogrow Policy.

Esempi di configurazione

Gli esempi seguenti mostrano configurazioni comuni di Autogrow Policy per diversi casi d'uso.

Politica conservativa per database di produzione

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

Archiviazione dei log con incrementi di crescita fissi

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: log-storage-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Politica minima con valori predefiniti

Quando si omettono `growthAmount` e `maxSize`, Trident utilizza i valori predefiniti (`10%` crescita, dimensione illimitata):

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: logs-policy
spec:
  usedThreshold: "85%"
```

Policy con un maxSize personalizzato e growthAmount predefinito

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: default-ga-policy
spec:
  usedThreshold: "70%"
  maxSize: "100Gi"
```

Crescita aggressiva con maxSize illimitata

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

Policy con percentuali frazionarie

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: precise-policy
spec:
  usedThreshold: "80.28%"
  growthAmount: "10.65%"
  maxSize: "100Gi"
```

NOTA Sono supportate le percentuali frazionarie. Se si specificano più di tre cifre decimali, Trident arrotonda il valore a tre cifre decimali.

NAS StorageClass con Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN StorageClass con crescita automatica

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
```

```
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Configura il dimensionamento automatico del volume ONTAP

Configura Trident per abilitare il dimensionamento automatico dei volumi ONTAP sui volumi FlexVol che supportano i volumi persistenti. Quando lo spazio utilizzato raggiunge una soglia configurata, ONTAP espande automaticamente il FlexVol fino a una dimensione massima. Questo fornisce capacità aggiuntiva per snapshot e overhead dello storage senza dover ridimensionare manualmente il FlexVol.

Il dimensionamento automatico del volume ONTAP aumenta solo il FlexVol sottostante; non modifica le dimensioni del PersistentVolumeClaim (PVC). Questo differisce da "Policy di crescita automatica di Trident", che espandono il PVC.

Trident applica le impostazioni di dimensionamento automatico del backend quando crea, clona o ridimensiona un volume e quando importa un volume che gestisce.

Requisiti

- Trident 26.10 o versioni successive.
- Uno dei seguenti driver:
  - `ontap-san` con iSCSI, NVMe o Fibre Channel Protocol (FCP).
  - `ontap-san-economy` con iSCSI.
- ONTAP on-premises:
  - Per i backend che utilizzano le API REST di ONTAP, è richiesta la versione 9.12.1 o successiva di ONTAP. I backend NVMe devono utilizzare le API REST di ONTAP.
  - Per i backend che utilizzano ONTAPI (ZAPI), qualsiasi versione di ONTAP supportata da Trident. Vedi "Requisiti Trident".

Limitazioni

- Il dimensionamento automatico del volume ONTAP non riduce i FlexVols.
- Il dimensionamento automatico è configurato per ogni backend. Crea backend separati per utilizzare impostazioni diverse per carichi di lavoro diversi.
- Il dimensionamento automatico del volume ONTAP non è supportato sui driver NAS ONTAP, sui sistemi ASA r2, su NetApp AFX o altri sistemi ONTAP disaggregati, né su Google Cloud NetApp Volumes. La creazione del backend non riesce con queste configurazioni.
- La replica TridentMirrorRelationship e le configurazioni MetroCluster non sono supportate con il dimensionamento automatico del volume ONTAP. Trident non impedisce queste configurazioni.

- Per `ontap-san-economy`, `maximumSize` deve essere una percentuale.
- Per `ontap-san`, un valore assoluto `maximumSize` funge da limite dimensionale. La creazione o l'espansione di un PVC non riesce se il FlexVol sottostante deve superare tale limite.
- Un `ontap-san-economy` FlexVol può contenere LUN per più PVC. Ogni operazione sul volume applica le impostazioni di dimensionamento automatico del backend corrente all'intero FlexVol. Se più backend condividono FlexVols, l'operazione più recente ne determina le impostazioni. Per `ontap-san-economy`, configura solo `mode: grow` e ometti `growThresholdPercent` e `maximumSize` così tutti i backend usano le stesse impostazioni predefinite.

Opzioni di dimensionamento automatico del volume ONTAP

Imposta queste opzioni nel blocco di primo livello `ontapAutosize` della configurazione del backend.

Opzione	Descrizione	Formato	Obbligatorio	Predefinito quando `ontapAutosize` è presente	Esempio
`mode`	Abilita o disabilita il dimensionamento automatico. `grow` abilita il dimensionamento automatico; `off` lo disabilita.	Stringa	No	`off`	`grow` o `off`
`growThresholdPercent`	Percentuale di spazio utilizzato alla quale ONTAP aumenta le dimensioni di FlexVol. I valori validi vanno da 2 a 99.	Intero	No	`85`	`90`
`maximumSize`	Dimensione massima di FlexVol, espressa come percentuale della sua dimensione o come dimensione assoluta. Le dimensioni assolute sono supportate solo per `ontap-san`.	Percentuale o quantità Kubernetes	No	`"20%"`	`"20%"` o `"2Ti"`

Le impostazioni predefinite si applicano quando il `ontapAutosize` blocco è presente. Se ometti il blocco, Trident non modifica le impostazioni di dimensionamento automatico su FlexVols.

Trident converte una percentuale `maximumSize` in una dimensione assoluta e la ricalcola ogni volta che ridimensioni il volume. Ad esempio, con `maximumSize: "20%"`, un FlexVol da 100 GiB può crescere fino a 120 GiB.

Configura il dimensionamento automatico del volume ONTAP

1. Aggiungi il `ontapAutosize` blocco alla configurazione del tuo backend:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
  namespace: trident
spec:
```

```
  version: 1
  storageDriverName: ontap-san
  backendName: ontap-san-autosize
  managementLIF: 10.0.0.1
  svm: svm_san
  sanType: iscsi
  ontapAutosize:
    mode: grow
    growThresholdPercent: 85
    maximumSize: "20%"
  credentials:
    name: backend-tbc-ontap-san-secret
```

2. Crea il backend:

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

3. Dopo aver eseguito il provisioning di un volume da questo backend, verifica le impostazioni di dimensionamento automatico applicate al volume Trident:

```
kubectl get tridentvolume <pv-name> -n trident \
  -o jsonpath='{.config.ontapAutosizeMode}
{.config.ontapAutosizeGrowThresholdPercent}
{.config.ontapAutosizeMaximumSize}'
```

Per la configurazione sopra descritta, l'output previsto è:

```
grow 85 20%
```

Aggiorna le impostazioni di dimensionamento automatico in un backend

Puoi aggiungere, modificare o rimuovere il blocco `ontapAutosize` aggiornando il backend. La modifica non aggiorna immediatamente i FlexVols esistenti:

- Per `ontap-san`, i nuovi volumi, i cloni e le importazioni gestite usano le nuove impostazioni. I volumi esistenti mantengono le impostazioni correnti, anche quando li ridimensioni.
- Per `ontap-san-economy`, il FlexVol condiviso riceve le nuove impostazioni la prossima volta che Trident crea, ridimensiona, clona o importa un volume al suo interno.

Esempi di configurazione

`ontap-san` con FCP e una dimensione massima assoluta

```
{
```

```
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_san",
  "username": "vsadmin",
  "password": "<password>",
  "sanType": "fcp",
  "ontapAutosize": {
    "mode": "grow",
    "growThresholdPercent": 90,
    "maximumSize": "2Ti"
  }
}
```

`ontap-san-economy`

Questo esempio imposta solo `mode`, quindi Trident usa i valori predefiniti per `growThresholdPercent` e `maximumSize`:

```
{
  "version": 1,
  "storageDriverName": "ontap-san-economy",
  "managementLIF": "10.0.0.1",
  "svm": "svm_san",
  "username": "vsadmin",
  "password": "<password>",
  "ontapAutosize": {
    "mode": "grow"
  }
}
```

Domande frequenti

Posso utilizzare il dimensionamento automatico del volume ONTAP con le policy di crescita automatica di Trident?

Sì. Una Trident Autogrow Policy espande il PVC e Trident ricalcola il massimo del dimensionamento automatico di ONTAP durante ogni ridimensionamento. Se usi un valore assoluto `maximumSize` con `ontap-san`, imposta il valore del criterio `maxSize` abbastanza basso da mantenere il FlexVol sottostante entro tale limite. Il FlexVol è più grande del PVC perché Trident aggiunge spazio per la riserva snapshot e i metadati.

Qual è la differenza tra omettere `ontapAutosize` e impostare `mode: off`?

Se ometti `ontapAutosize`, Trident non modifica le impostazioni di dimensionamento automatico sui FlexVols di ONTAP. Se imposti `mode: off`, Trident disabilita il dimensionamento automatico sui FlexVols che crea o aggiorna, inclusi i cloni che hanno ereditato le impostazioni di dimensionamento automatico dall'origine.

Perché l'espansione del PVC non è riuscita con un `ontapAutosize maximum … is below required FlexVol size` errore?

Il volume utilizza un valore assoluto `maximumSize`, e il PVC espanso richiede un FlexVol più grande. Un `ontap-san` volume mantiene la dimensione massima con cui è stato creato, quindi l'aggiornamento del backend non modifica l'impostazione sui volumi esistenti. Per i nuovi volumi, aumenta `maximumSize` o usa una percentuale.

Perché la creazione del volume non è riuscita a causa di un errore di dimensionamento automatico?

ONTAP ha rifiutato le impostazioni di dimensionamento automatico e gli eventi PVC includono il motivo fornito da ONTAP. Ad esempio, la dimensione massima potrebbe superare la dimensione massima di FlexVol supportata dal sistema ONTAP. Riduci `maximumSize` nel backend; Trident usa il valore aggiornato la volta successiva che tenta di creare un volume.

Informazioni correlate

- "Espansione automatica del volume con Trident Autogrow Policies"
- "Riferimento ai comandi di ONTAP: volume dimensionamento automatico"

Gestisci le policy di Autogrow

Dopo che crei le Autogrow Policies, puoi visualizzarle, aggiornarle ed eliminarle. Puoi anche monitorare quali volumi usano una determinata policy.

Visualizza le policy di crescita automatica

Elenca tutte le politiche

Usa `kubectl` per elencare tutte le Autogrow Policies nel tuo cluster:

```
kubectl get tridentautogrowpolicy
```

In alternativa, usa `tridentctl`:

```
tridentctl get autogrowpolicy
```

Visualizza dettagli della polizza

Per visualizzare le specifiche complete e lo stato di una policy:

```
kubectl describe tridentautogrowpolicy production-db-policy
```

Per visualizzare una policy con i volumi associati in formato YAML:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Aggiorna una policy di Autogrow

È possibile modificare una policy esistente per cambiarne la soglia, l'entità della crescita o la dimensione massima. Le modifiche hanno effetto immediato per tutti i volumi che utilizzano la policy.

IMPORTANTE	Le modifiche influiscono su tutti i volumi che utilizzano la policy. Quando possibile, testa prima le modifiche in un ambiente non di produzione.

Fasi

1. Modifica la policy:

```
kubectl edit tridentautogrowpolicy production-db-policy
```

2. Modifica i `spec` campi secondo necessità:

```
spec:
  usedThreshold: "75%"     # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

3. Salva ed esci. Le modifiche hanno effetto immediato.

Considerazioni sull'aggiornamento

- **Effetto immediato:** Tutti i volumi che utilizzano la policy adottano i nuovi parametri alla successiva valutazione della crescita.
- **Non è necessario riavviare il volume:** Le modifiche si applicano alla prossima operazione di crescita.
- **Test first:** convalidare le modifiche in un ambiente non di produzione quando possibile.
- **Comunica le modifiche:** Notifica ai team quando modifichi le policy condivise.

Elimina un criterio Autogrow

Le policy di crescita automatica utilizzano la protezione del finalizzatore per impedire la cancellazione accidentale mentre i volumi le utilizzano attivamente.

Fasi

1. Elimina la policy:

```
kubectl delete tridentautogrowpolicy production-db-policy
```

2. Se i volumi utilizzano ancora la policy, l'eliminazione entra in uno stato `Deleting`. Verifica quali volumi sono interessati:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

3. Rimuovi il criterio da ogni volume interessato. Scegli una delle seguenti opzioni:

   - **Opzione A: Disattivare esplicitamente l'autogrow** impostando l'annotazione su `"none"`:

     ```
     kubectl annotate pvc <pvc-name> \
       trident.netapp.io/autogrowPolicy="none" \
       --overwrite
     ```

   - **Opzione B: Rimuovi completamente l'annotazione:**

     ```
     kubectl annotate pvc <pvc-name> \
       trident.netapp.io/autogrowPolicy-
     ```

Comportamento di eliminazione

Scenario	Comportamento
Nessun volume utilizza la policy	La policy viene eliminata immediatamente.
I volumi utilizzano la policy	La policy entra `Deleting` nello stato. Un finalizzatore blocca il completamento finché tutti i volumi non vengono rimossi.
Tutti i volumi vengono rimossi dalla policy	I finalizzatori vengono rimossi e la policy viene eliminata.

Monitorare l'utilizzo della policy Autogrow

Verificare i volumi utilizzando una policy

```
tridentctl get autogrowpolicy production-db-policy -o json | jq '.volumes'
```

Scopri quale policy utilizza un volume

```
kubectl get pvc database-pvc -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Monitorare gli eventi delle policy

```
kubectl get events --field-selector
involvedObject.kind=TridentAutogrowPolicy
```

Protocolli supportati

Autogrow supporta i seguenti protocolli di archiviazione:

- NFS
- ISCSI
- FCP
- NVMe

NOTA	Per i volumi SAN, se la `growthAmount` configurata è pari o inferiore a 50 MiB, Trident aumenta automaticamente la quantità di crescita a 51 MB per l'operazione di ridimensionamento, a condizione che la dimensione risultante non superi `maxSize`.

Backend supportati

Autogrow supporta i backend ONTAP. Autogrow supporta anche Google Cloud NetApp Volumes (GCNV) sia per i volumi NAS che SAN.

Limitazioni note

- **ONTAP NVMe raw block volumes:** I volumi creati con versioni di ONTAP precedenti alla 9.16.1 non supportano l'autogrow.
- **Volumi esistenti (implementazioni brownfield):** L'espansione automatica potrebbe non funzionare per i volumi esistenti anche se è stata applicata una Autogrow Policy valida. Questa limitazione è dovuta a una migrazione in corso delle pubblicazioni dei volumi. Per verificare che la migrazione sia completa, controlla i log del controller Trident per messaggi `"Migration completed"`.

Domande frequenti

Quando Trident valuta la soglia?

Trident monitora costantemente l'utilizzo del volume. Quando la capacità utilizzata supera il `usedThreshold`, Trident crea una richiesta di ridimensionamento interna ed espande il volume della `growthAmount` configurata.

Ad esempio, questa policy attiva l'espansione all'80% della capacità e aumenta il volume del 10% ogni volta, fino a un massimo di 500 GiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

Posso applicare una policy dopo che i volumi sono già stati forniti?

Sì. È possibile creare una Autogrow Policy in qualsiasi momento e applicarla ai PVC esistenti aggiungendo o aggiornando l' `trident.netapp.io/autogrowPolicy`annotazione. Non è necessario ricreare il PVC o il StorageClass.

Per applicare una policy a un PVC esistente:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Per applicare una policy a un'istanza esistente di StorageClass:

```
kubectl annotate storageclass ontap-gold \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Cosa succede se imposto una politica di crescita automatica sia sulla StorageClass che sul PVC?

L'annotazione PVC ha sempre la precedenza. Se un PVC ha l'annotazione `trident.netapp.io/autogrowPolicy`, Trident utilizza quel valore indipendentemente da quanto specificato in StorageClass. Consultare "Precedenza delle policy" per i dettagli.

Ad esempio, dato questo StorageClass:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

E questo PVC che annulla la politica StorageClass:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
```

```
  storageClassName: ontap-gold
```

Trident usa `logs-policy` per `database-pvc`, non `standard-agp`.

Come faccio a disattivare l'autogrow per un volume specifico?

Imposta l'annotazione PVC su `"none"`. Questa impostazione sovrascrive qualsiasi policy a livello di StorageClass per quel volume:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

Puoi verificare che autogrow sia disabilitato:

```
kubectl get pvc <pvc-name> -o jsonpath
='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Output previsto

```
none
```

Cosa succede quando un volume raggiunge maxSize?

Trident interrompe l'espansione del volume. Non crea ulteriori richieste di ridimensionamento per quel volume, anche se l'utilizzo continua ad aumentare oltre il `usedThreshold`.

Ad esempio, con questa policy, Trident interrompe la crescita del volume dopo che raggiunge i 100 GiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: capped-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Per consentire una crescita illimitata, ometti `maxSize` o impostalo su `0`:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: unlimited-policy
```

```
spec:
  usedThreshold: "85%"
  growthAmount: "10%"
```

Posso modificare una policy senza riavviare i volumi?

Sì. Quando aggiorni una policy, tutti i volumi che utilizzano quella policy adottano i nuovi parametri alla successiva valutazione della crescita. Non è necessario riavviare i volumi.

Per aggiornare una policy in vigore:

```
kubectl edit tridentautogrowpolicy production-db-policy
```

Modifica i campi secondo necessità:

```
spec:
  usedThreshold: "75%"     # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

Salva ed esci. Verifica la policy aggiornata:

```
kubectl get tridentautogrowpolicy production-db-policy
```

Output previsto

```
NAME                    USED THRESHOLD   GROWTH AMOUNT   STATE
production-db-policy    75%              20%             Success
```

Perché la mia policy è in stato Failed?

Uno `Failed` stato indica che la specifica della policy contiene errori di convalida. Esegui il seguente comando per visualizzare i dettagli dell'errore:

```
kubectl describe tridentautogrowpolicy <policy-name>
```

Le cause più comuni includono un `usedThreshold` non valido (deve essere compreso tra 1 e 99%), un `growthAmount` che supera `maxSize`, o un formato di quantità Kubernetes non valido. Correggere la specifica e riapplicarla:

```
kubectl apply -f autogrow-policy.yaml
```

Perché non posso eliminare una policy?

Le policy utilizzano la protezione del finalizzatore. Se i volumi utilizzano ancora la policy, l'eliminazione entra in uno stato `Deleting` e attende finché non rimuovi tutti i volumi dalla policy.

Identificare i volumi interessati:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Quindi rimuovere l'annotazione da ogni PVC:

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Dopo aver rimosso tutti i volumi, Trident rilascia il finalizzatore ed elimina la policy.

Quali backend sono supportati da autogrow?

Autogrow supporta i backend ONTAP. Supporta inoltre Google Cloud NetApp Volumes (GCNV) sia per i volumi NAS che SAN.

Autogrow supporta i protocolli NFS, iSCSI, FCP e NVMe. I volumi a blocchi raw NVMe richiedono ONTAP 9.16.1 o versioni successive.

I volumi esistenti nelle implementazioni brownfield potrebbero richiedere il completamento della migrazione della pubblicazione dei volumi prima che l'espansione automatica abbia effetto. Per verificare lo stato della migrazione, controlla i log del controller Trident:

```
kubectl logs -l app=trident-controller -n trident | grep "Migration
completed"
```

I seguenti esempi di StorageClass mostrano l'autogrow configurato per i backend NAS e SAN:

Backend NAS

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
```

```
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

Backend SAN

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Qual è la quantità minima di crescita per i volumi SAN?

Per i volumi SAN, la crescita minima effettiva è di 51 MB. Se si configura un `growthAmount` di 50 MiB o meno, Trident aumenta automaticamente la crescita a 51 MB per l'operazione di ridimensionamento.

Ad esempio, questa policy imposta una `growthAmount` di `"40Mi"`, ma Trident applica una crescita di 51 MB per qualsiasi volume SAN che la utilizza:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-minimal-policy
spec:
  usedThreshold: "85%"
  growthAmount: "40Mi"
  maxSize: "100Gi"
```

Per evitare questa regolazione automatica, impostare `growthAmount` su un valore maggiore di 50 MiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-policy
spec:
  usedThreshold: "85%"
```

```
    growthAmount: "100Mi"
    maxSize: "500Gi"
```

Importa i volumi Trident

È possibile importare volumi di archiviazione esistenti come PV Kubernetes utilizzando `tridentctl import` o creando un Persistent Volume Claim (PVC) con annotazioni di importazione Trident.

Panoramica e considerazioni

È possibile importare un volume in Trident in:

- 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.

- Trident può importare solo volumi ONTAP di tipo RW (lettura-scrittura). I volumi di tipo DP (data Protection) sono volumi di destinazione SnapMirror. Interrompere la relazione di mirroring prima di importare il volume in Trident.
- Si consiglia di importare volumi senza connessioni attive. Per importare un volume utilizzato attivamente, clonare il volume ed eseguire l'importazione.

ATTENZIONE	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.

- Sebbene `StorageClass` debba essere specificato su un PVC, 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 la reclaim policy della Storage Class è `delete`, Trident elimina il volume di storage quando il PV viene eliminato.
- Per impostazione predefinita, Trident gestisce il PVC e rinomina il FlexVol volume e la LUN o qtree sul backend. Puoi passare il `--no-manage` flag per importare un volume non gestito e il `--no-rename` flag per mantenere il nome del volume.

- `--no-manage`* - Se usi il `--no-manage` flag, Trident non esegue alcuna operazione aggiuntiva sul PVC o sul PV per il ciclo di vita degli oggetti. Il volume di archiviazione non viene eliminato quando si elimina il PV e anche altre operazioni come la clonazione del volume e il ridimensionamento del volume vengono ignorate.
- `--no-rename`* - Se usi il `--no-rename`flag, Trident mantiene il nome del volume esistente durante l'importazione dei volumi e gestisce il ciclo di vita dei volumi. Questa opzione è supportata solo per i `ontap-nas`, `ontap-san` (inclusi i sistemi ASA r2), `ontap-san-economy` e `ontap-nas-economy` (importazione qtree) driver.

SUGGERIMENTO	Queste opzioni sono utili se si desidera utilizzare Kubernetes per carichi di lavoro containerizzati ma si desidera gestire il ciclo di vita del volume di archiviazione 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 importare un volume utilizzando `tridentctl import` oppure creando un PVC con annotazioni di importazione Trident.

NOTA	Se si utilizzano annotazioni PVC, non è necessario scaricare o utilizzare `tridentctl` per importare il volume.

Utilizzo di tridentctl

Fasi

1. Crea un file PVC (ad esempio, `pvc.yaml`) che verrà utilizzato per creare il PVC. Il file PVC dovrebbe includere `name`, `namespace`, `accessModes` e `storageClassName`. Facoltativamente, puoi specificare `unixPermissions` nella definizione del 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
   ```

NOTA	Includere solo i parametri obbligatori. Parametri aggiuntivi come il nome del PV o la dimensione del volume possono causare il fallimento del comando di importazione.

2. Utilizzare il `tridentctl import` comando per specificare il nome del backend Trident contenente il volume e il nome che identifica in modo univoco il volume nello storage (ad esempio: ONTAP FlexVol,

Element Volume). IL `-f` L'argomento è necessario per specificare il percorso del file PVC.

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

Utilizzo delle annotazioni PVC

Fasi

1. Creare un file PVC YAML (ad esempio, `pvc.yaml`) con le annotazioni di importazione Trident richieste. Il file PVC deve includere:
   - `name` and `namespace` nei metadati
   - `accessModes`, `resources.requests.storage`, e `storageClassName` nelle specifiche
   - Annotazioni:
     - `trident.netapp.io/importOriginalName`: Nome volume sul backend
     - `trident.netapp.io/importBackendUUID`: UUID del backend in cui esiste il volume
     - `trident.netapp.io/notManaged` (*Facoltativo*): Impostare su `"true"` per i volumi non gestiti. Predefinito è `"false"`.

       Di seguito è riportato un esempio di specifica per l'importazione di un volume gestito:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. Applica il file PVC YAML al tuo cluster Kubernetes:

```
kubectl apply -f <pvc-file>.yaml
```

Trident importerà automaticamente il volume e lo assocerà al PVC.

Esempi

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

NAS ONTAP e NAS FlexGroup ONTAP

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

NOTA

- Per importare i qtree con il `ontap-nas-economy` driver, consulta Importa qtree NFS. Quando Trident importa un qtree, importa anche il FlexVol padre come bucket che contiene il qtree. Devi importare i qtree singolarmente.
- Il `ontap-nas` e. `ontap-nas-flexgroup` i driver non consentono nomi di volumi duplicati.

Ogni volume creato con il `ontap-nas` driver è un FlexVol volume nel cluster ONTAP. L'importazione dei volumi FlexVol con il `ontap-nas` driver funziona allo stesso modo. È possibile importare come PVC i volumi FlexVol già presenti in un cluster ONTAP `ontap-nas`. Analogamente, i FlexGroup vol possono essere importati come `ontap-nas-flexgroup` PVC.

Esempi di ONTAP NAS utilizzando tridentctl

Gli esempi seguenti mostrano come importare volumi gestiti e non gestiti utilizzando `tridentctl`.

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 l' `--no-manage`argomento, 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   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+---------+---------+
```

Esempi ONTAP NAS che utilizzano annotazioni PVC

I seguenti esempi mostrano come importare volumi gestiti e non gestiti utilizzando annotazioni PVC.

Volume gestito

Il seguente esempio importa un volume da 1GiB `ontap-nas` denominato `ontap_volume1` dal backend `81abcb27-ea63-49bb-b606-0a5315ac5f21` con modalità di accesso RWO impostata tramite annotazioni PVC:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

Volume non gestito

Il seguente esempio importa 1Gi `ontap-nas` volume denominato `ontap-volume2` dal backend `34abcb27-ea63-49bb-b606-0a5315ac5f34` con modalità di accesso RWO impostata tramite annotazioni PVC:

```
kind: PersistentVolumeClaim
```

```
apiVersion: v1
metadata:
  name: <umanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-
0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

Importa qtree

NOTA	L'importazione qtree è una funzionalità Tech Preview in Trident 26.10. Le funzionalità Tech Preview non sono supportate per l'utilizzo in produzione.

Puoi importare qtree ONTAP esistenti in Trident utilizzando il driver `ontap-nas-economy`. Identifica il qtree in base al nome del volume FlexVol e al nome del qtree nel formato `flexvol/qtree`, insieme all'UUID del backend che lo contiene. Le considerazioni generali sull'importazione in Panoramica e considerazioni si applicano anche all'importazione dei qtree.

Importa un qtree usando tridentctl

Specifica il qtree nel formato `flexvol/qtree` come nome del volume. Il seguente esempio importa il qtree `qtree1` dal volume FlexVol `flexvol1` sul backend `ontap_nas_economy`:

```
tridentctl import volume ontap_nas_economy flexvol1/qtree1 -f <path-to-
pvc-file>
```

Per importare il qtree senza rinominarlo, aggiungi il `--no-rename` flag. Per importare il qtree senza che Trident ne gestisca il ciclo di vita, aggiungi il `--no-manage` flag.

Importa un qtree usando le annotazioni PVC

Crea un PVC con le seguenti annotazioni:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
```

```
  annotations:
    trident.netapp.io/importOriginalName: "flexvol1/qtree1"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

Per mantenere i nomi esistenti del volume FlexVol e del qtree, aggiungi l' `trident.netapp.io/importNoRename: "true"`annotazione. Per importare il qtree come non gestito, aggiungi l' `trident.netapp.io/notManaged: "true"`annotazione.

Denominazione e comportamento gestionale

Importazione gestita (impostazione predefinita)

Trident rinomina il qtree in modo che corrisponda al nome del PVC e rinomina il FlexVol volume secondo la `ontap-nas-economy` convenzione di naming. Il percorso di giunzione non cambia.

`--no-rename`

Trident mantiene i nomi esistenti dei FlexVol volume e dei qtree. I nomi devono già seguire le convenzioni di naming di Trident, altrimenti Trident non può gestire le operazioni di eliminazione o montaggio del qtree.

`--no-manage`

Trident fa riferimento al qtree ma non ne gestisce il ciclo di vita. I nomi devono seguire le convenzioni di naming di Trident e Trident non utilizza né gestisce policy di esportazione automatica per il qtree.

Comportamento della policy di esportazione

Se il backend utilizza le policy di esportazione automatica, il qtree importato utilizza il modello di policy di esportazione per qtree, in cui ogni qtree ha la propria policy di esportazione. Trident assegna al qtree una policy di esportazione vuota al momento dell'importazione. Al momento della pubblicazione, Trident aggiorna il volume FlexVol alla policy di esportazione per backend se l'esportazione automatica è abilitata e la policy di esportazione FlexVol esistente è `default` o vuota.

AVVERTENZA	L'aggiornamento della policy di esportazione del volume FlexVol può interrompere altri qtree nello stesso volume FlexVol che sono attualmente montati. Importa i qtree solo quando i qtree nel volume FlexVol sono smontati.
ATTENZIONE	Non indirizzare due cluster Kubernetes allo stesso SVM attivo e non importare un qtree che un altro cluster gestisce ancora. Se entrambi i cluster gestiscono lo stesso volume, operazioni in conflitto possono corrompere o distruggere i dati.

ONTAP SAN

Trident supporta l'importazione di volumi utilizzando `ontap-san` (iSCSI, NVMe/TCP e FC) e `ontap-san-economy` conducenti.

Trident può importare volumi ONTAP SAN FlexVol che contengono un singolo LUN. Ciò è coerente con il

`ontap-san` driver, che crea un FlexVol volume per ogni PVC e un LUN all'interno del FlexVol volume. Trident importa il FlexVol volume e lo associa alla definizione PVC. Trident può importare `ontap-san-economy` volumi che contengono più LUN.

I seguenti esempi mostrano come importare volumi gestiti e non gestiti:

Volume gestito

Per i volumi gestiti, Trident rinomina FlexVol volume nel `pvc-<uuid>` formato e il LUN all'interno di FlexVol volume in `lun0`.

Nell'esempio seguente viene importato il `ontap-san-managed` FlexVol volume presente sul `ontap_san_default` backend:

```
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
```

Esempi di ONTAP SAN-economy

I seguenti esempi mostrano come importare volumi gestiti e non gestiti per il backend. `ontap-san-economy`

Volume gestito

Quando si importa un volume gestito, Trident assume la proprietà del FlexVol e lo rinomina. È necessario tenere conto di questa ridenominazione quando si importano più LUN dallo stesso FlexVol.

Il seguente esempio importa `lun1` dal FlexVol `toimport` come volume gestito denominato `vol-managed-saneco`:

```
tridentctl import volume vol-managed-saneco toimport/lun1 -f
import1.yaml
```

Dopo l'importazione `lun1`, Trident rinomina il FlexVol (ad esempio, in `trident_lun_pool_xyz`). Per importare ulteriori LUN dallo stesso FlexVol, utilizzare il nuovo nome FlexVol:

```
tridentctl import volume vol-managed-saneco trident_lun_pool_xyz/lun2
-f import2.yaml
```

NOTA Il `ontap-san-economy` backend importa un LUN alla volta. Puoi automatizzare più importazioni utilizzando uno script.

Volume non gestito

Quando si importa un volume non gestito, Trident non assume la proprietà del FlexVol. Tuttavia, il FlexVol e la LUN devono seguire le convenzioni di denominazione di Trident.

Formato di denominazione FlexVol

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- `STORAGEPREFIX` è il valore di `storagePrefix` nella configurazione del backend. Il valore predefinito è `trident`.
- `RANDOMSTRING` è una qualsiasi stringa tu scelga.

Requisito di denominazione LUN

Il LUN deve essere denominato `lun0`.

Esempio

Se il tuo `storagePrefix` è `xyz`, il full path al LUN è:

```
trident_lun_pool_xyz_randomstring/lun0
```

Elemento

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

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

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

Azure NetApp Files

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

NOTA

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`.

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

Google Cloud NetApp Volumes

Trident supporta l'importazione di volumi utilizzando il `google-cloud-netapp-volumes` driver.

Il seguente esempio importa un volume su backend `backend-tbc-gcnv1` con il volume `testvoleasiaeast1`.

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
```

```
+--------+---------+
```

Nell'esempio seguente viene importato un `google-cloud-netapp-volumes` volume quando nella stessa regione sono presenti due volumi:

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

Personalizza i nomi e le etichette dei volumi di Trident

Con Trident, è possibile assegnare nomi e etichette significativi ai volumi creati. Questo ti aiuta a identificare e mappare facilmente i volumi alle rispettive risorse Kubernetes (PVC). È inoltre possibile definire modelli di backend per la creazione di nomi di volumi personalizzati ed etichette personalizzate; i volumi creati, importati o clonati aderiranno ai modelli.

Prima di iniziare

Nomi di volumi ed etichette personalizzabili supportano:

- Operazioni di creazione, importazione e cloning del volume.
- Nel caso del `ontap-nas-economy` driver, solo il nome del volume Qtree è conforme al modello di nome.
- Nel caso del `ontap-san-economy` driver, solo il nome LUN è conforme al modello di nome.

Limitazioni

- I nomi dei volumi personalizzati sono compatibili solo con i driver ONTAP locali.
- Le etichette personalizzate sono supportate solo per `ontap-san`, `ontap-nas`, E `ontap-nas-flexgroup` conducenti.

- I nomi di volume personalizzati non si applicano ai volumi esistenti.

Comportamenti chiave dei nomi di volume personalizzabili

- Se si verifica un errore a causa di una sintassi non valida in un modello di nome, la creazione del backend non riesce. Tuttavia, se l'applicazione modello non riesce, il volume verrà denominato in base alla convenzione di denominazione esistente.
- Il prefisso di archiviazione non è applicabile quando un volume viene nominato utilizzando un modello di nome dalla configurazione backend. Qualsiasi valore di prefisso desiderato può essere aggiunto direttamente al modello.

Esempi di configurazione backend con modello di nome ed etichette

I modelli con nomi personalizzati possono essere definiti a livello di root e/o pool.

Esempio di livello root

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.Requ
estName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

Esempio di livello pool

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
```

```
    {
      "labels": {
        "labelname": "label1",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels": {
        "cluster": "label2",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

Esempi di modelli di nome

Esempio 1:

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

Esempio 2:

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

Punti da considerare

1. Nel caso di importazioni di volumi, le etichette vengono aggiornate solo se il volume esistente presenta etichette in un formato specifico. Ad esempio: `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. Nel caso di importazioni di volumi gestiti, il nome del volume segue il modello di nome definito al livello principale nella definizione di backend.
3. Trident non supporta l'uso di un operatore di sezione con il prefisso di memorizzazione.
4. Se i modelli non generano nomi di volume univoci, Trident aggiungerà alcuni caratteri casuali per creare

nomi di volume univoci.

5. Se il nome personalizzato per un volume economico NAS supera i 64 caratteri di lunghezza, Trident denominerà i volumi in base alla convenzione di denominazione esistente. Per tutti gli altri driver ONTAP, se il nome del volume supera il limite del nome, il processo di creazione del volume non riesce.

Condividi un volume NFS Trident tra namespace

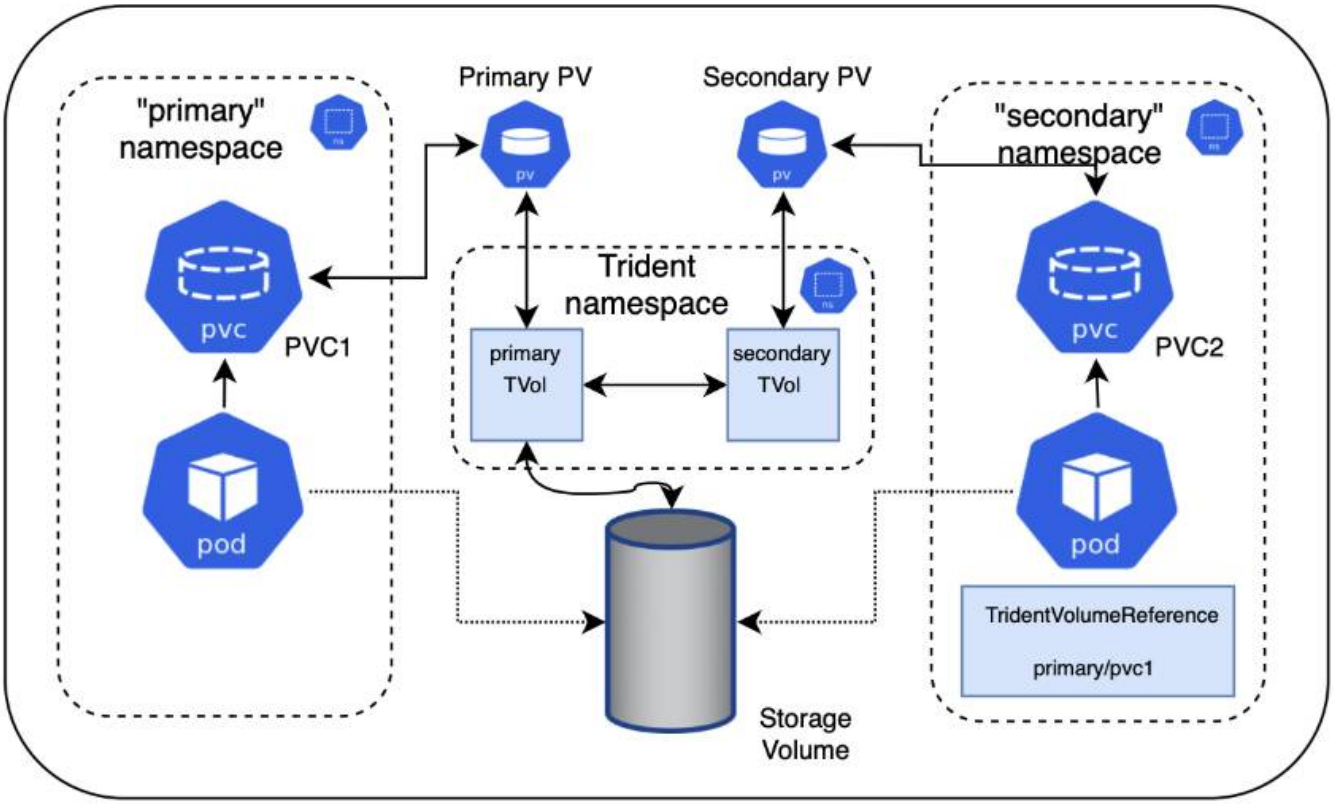
Utilizzando Trident, è possibile creare un volume in un namespace primario e condividerlo in uno o più namespace secondari.

Caratteristiche

TridentVolumeReference CR consente di condividere in modo sicuro i volumi NFS ReadWriteMany (RWX) in uno o più namespace 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
```

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

NOTA

- 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. **Amministratore del cluster:** assicurarsi che sia presente il corretto RBAC per concedere l'autorizzazione al proprietario dello spazio dei nomi di destinazione per creare il CR 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
```

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

Risultati

Trident legge l' `shareFromPVC`annotazione sul PVC di destinazione e crea il PV di destinazione come volume subordinato senza una risorsa di storage propria 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. Trident rimuoverà l'accesso al volume sul namespace di origine e manterrà l'accesso agli altri namespace che condividono il volume. Quando tutti gli spazi dei nomi che fanno riferimento al volume vengono rimossi, Trident elimina il volume.

Utilizzare `tridentctl get` per eseguire query sui volumi subordinati

Utilizzando l'[`tridentctl`utility, è possibile eseguire il `get`comando per ottenere volumi subordinati. Per ulteriori informazioni, consultare `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

- 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:

- Guarda la demo su "NetAppTV".

Clona i volumi Trident tra namespace

Utilizzando Trident, puoi creare nuovi volumi utilizzando volumi esistenti o volumesnapshot da un namespace diverso all'interno dello stesso cluster Kubernetes.

Prerequisiti

Prima di clonare i volumi, verificare che i backend di origine e di destinazione siano dello stesso tipo e abbiano la stessa classe di storage.

NOTA	La clonazione tra spazi dei nomi è supportata solo per `ontap-san` E `ontap-nas` driver di archiviazione. I cloni di sola lettura non sono supportati.

Avvio rapido

Il cloning dei volumi può essere configurato in pochi passaggi.



Configurare il PVC di origine per clonare il volume

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



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.



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.



Creare il PVC clone nello spazio dei nomi di destinazione

Il proprietario dello spazio dei nomi di destinazione crea PVC per clonare il PVC dallo spazio dei nomi di origine.

Configurare gli spazi dei nomi di origine e di destinazione

Per garantire la sicurezza, il cloning dei volumi negli spazi dei nomi richiede collaborazione e azione da parte 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. **Proprietario dello spazio dei nomi di origine:** creare il PVC (`pvc1`nello spazio dei (`namespace1`nomi di origine) che concede il permesso di condividere con lo spazio dei nomi di destinazione (`namespace2`) utilizzando l' `cloneToNamespace`annotazione.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
```

```
  annotations:
    trident.netapp.io/cloneToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident crea il PV e il suo volume di storage di backend.

NOTA

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

2. **Amministratore del cluster:** assicurarsi che sia presente il corretto RBAC per concedere l'autorizzazione al proprietario dello spazio dei nomi di destinazione per creare il CR TridentVolumeReference nello spazio dei nomi di destinazione(`namespace2`).
3. **Proprietario dello spazio dei nomi di destinazione:** Crea un CR TridentVolumeReference nello spazio dei nomi di destinazione che faccia riferimento al PVC 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:** creare un PVC (`namespace2`)(`pvc2` nello spazio dei nomi di destinazione utilizzando la `cloneFromPVC` o `cloneFromSnapshot`, e `cloneFromNamespace` le annotazioni per designare il PVC di origine.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
```

```
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Limitazioni

- Per i PVC forniti utilizzando driver ONTAP-nas-Economy, i cloni di sola lettura non sono supportati.

Replica dei volumi con SnapMirror

Trident supporta le relazioni di mirroring tra un volume di origine su un cluster e il volume di destinazione sul cluster in peering per la replica dei dati per il disaster recovery. È possibile utilizzare una definizione di risorsa personalizzata (CRD) con namespace, denominata Trident Mirror Relationship (TMR), 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

- **Trident**: Trident versione 22,10 o successiva deve esistere su entrambi i cluster Kubernetes di origine e 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.

  A partire da ONTAP 9.10.1, tutte le licenze vengono fornite come file di licenza NetApp (NLF), che è un singolo file che abilita più funzioni. Per ulteriori informazioni, fare riferimento "Licenze incluse con ONTAP ONE" a.

  NOTA È supportata solo la protezione asincrona SnapMirror.

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.

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

- **Trident e SVM**: Le SVM remote in peering devono essere disponibili per Trident nel cluster di destinazione.

Driver supportati

NetApp Trident supporta la replicazione dei volumi con la tecnologia NetApp SnapMirror utilizzando classi di archiviazione supportate dai seguenti driver: **`ontap-nas` : NFS** `ontap-san` : iSCSI **`ontap-san` : FC** `ontap-san` : NVMe/TCP (richiede almeno la versione ONTAP 9.15.1)

NOTA	La replicazione dei volumi tramite SnapMirror non è supportata per i sistemi ASA r2. Per informazioni sui sistemi ASA r2, vedere "Informazioni sui sistemi di storage ASA R2" .

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
```

Trident recupera le informazioni interne per il volume e lo stato di protezione dei dati (DP) corrente del volume, quindi compila 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
  replicationPolicy: MirrorAllSnapshots
  replicationSchedule: 5min
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Il TMR `spec` ha quattro campi di primo livello: `state`, che imposta lo stato di relazione desiderato (vedi Stati di replica dei volumi); `volumeMappings`, che associa i PVC locali ai relativi handle di volume remoti; e i campi facoltativi `replicationPolicy` e `replicationSchedule` descritti di seguito.

I campi `replicationPolicy` e `replicationSchedule` sono facoltativi. Imposta `replicationPolicy` sul nome di un criterio asincrono di SnapMirror su ONTAP e `replicationSchedule` sul nome di una pianificazione ONTAP. Se ometti uno dei due campi, Trident utilizza il criterio di replica o la pianificazione corrispondente configurata sul backend. Imposta questi campi quando crei la relazione; non puoi modificare il criterio o la pianificazione dopo che la relazione è stata stabilita. Per utilizzare impostazioni diverse, elimina e ricrea il TMR.

Trident crea e inizializza una relazione di SnapMirror utilizzando la policy di replica e la pianificazione configurate.

1. 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
```

Trident verificherà la presenza del CRD TridentMirrorRelationship e non riuscirà a creare il volume se la relazione non esiste. Se la relazione esiste, Trident garantirà il posizionamento del nuovo FlexVol volume in una SVM a cui viene eseguito il peering con la SVM remota definita nella 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 a Trident 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, crea il TridentTridentSnapshotInfoCR per ottenere i dettagli interni.

   Esempio

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

   a. Sul cluster Kubernetes secondario, aggiorna il campo *spec.state* del CR *TridentMirrorRelationship* a *promoted* e *spec.volumeMappings[].promotedSnapshotHandle* al internalName dello snapshot.

3. 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, aggiorna il campo *spec.state* di TridentMirrorRelationship in `reestablished`.

Operazioni supplementari

Trident 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)

Trident verifica la presenza di PVC replicati e rilascia il rapporto 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 Trident completi l'eliminazione. Se la TMR selezionata per l'eliminazione è già nello stato *promosso*, non esiste alcuna relazione di mirroring e la TMR verrà rimossa e Trident 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 del SnapMirror senza che Trident 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*.

Utilizzare la topologia CSI

Trident può creare e collegare in modo selettivo i 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, Trident utilizza la topologia CSI.

SUGGERIMENTO Scopri di più sulla funzionalità topologia CSI "qui".

Kubernetes offre due esclusive modalità di binding del volume:

- Con `VolumeBindingMode` impostato su `Immediate`, Trident crea il volume senza alcuna conoscenza 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 adatta 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.

NOTA 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 avere etichette che introducano la consapevolezza della topologia `topology.kubernetes.io/region` e `topology.kubernetes.io/zone`). L' `topology.kubernetes.io/region`etichetta è obbligatoria per il provisioning con riconoscimento della topologia in Trident. Queste etichette devono essere presenti sui nodi prima dell'installazione di Trident. Se aggiungi o modifichi le etichette di topologia dopo l'installazione, è sufficiente riavviare i pod dei nodi di Trident perché Trident rilevi le modifiche.

```
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 Trident possono essere progettati per eseguire il provisioning selettivo dei volumi in base alle zone di disponibilità. Ogni backend può portare un blocco opzionale `supportedTopologies` che rappresenta un elenco di zone e regioni 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

```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"
    }
  ]
}
```

NOTA `supportedTopologies` viene utilizzato per fornire un elenco di aree 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, Trident crea 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
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
```

```
  - labels:
      workload: dev
    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 StorageClass fornita sopra, `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. E, `allowedTopologies` fornisce le zone e la regione da utilizzare. `netapp-san-us-east1`StorageClass crea PVC sul `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: null
name: pvc-san
spec: null
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"

Lavora con gli snapshot di Trident

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 Trident, importare uno snapshot creato al di fuori di 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` , `azure-netapp-files` , E `google-cloud-netapp-volumes` conducenti.

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.

NOTA	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".
   - `driver`Indica il driver 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.

ATTENZIONE	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

Trident supporta l' "Processo Snapshot con pre-provisioning di Kubernetes" per consentire all'amministratore del cluster di creare un `VolumeSnapshotContent` oggetto e importare snapshot creati all'esterno di Trident.

Prima di iniziare

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. Viene avviato il flusso di lavoro dello snapshot in 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>` in `snapshotHandle`. si tratta delle uniche informazioni fornite a Trident dallo snap-over esterno nella `ListSnapshots` chiamata.

     NOTA 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>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

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 il nuovo creato ed esegue `ListSnapshots` la `VolumeSnapshotContent` chiamata. Trident crea la `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
```

NOTA	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

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

Trident supporta il ripristino degli snapshot su `ontap-san` , `ontap-san-economy` , `ontap-nas` , `ontap-nas-flexgroup` , `azure-netapp-files` , `google-cloud-netapp-volumes` , E `solidfire-san` conducenti.

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`.

   NOTA Il TASR CR deve trovarsi in uno spazio dei nomi in cui esistono PVC e VS.

   ```
   cat tasr-pvc1-snapshot.yaml
   ```

   ```
   apiVersion: trident.netapp.io/v1
   kind: TridentActionSnapshotRestore
   metadata:
     name: trident-snap
     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/trident-snap created
   ```

Risultati

Trident ripristina i dati dalla snapshot. È possibile verificare lo stato di ripristino dello snapshot:

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    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: ""
```

NOTA

- Nella maggior parte dei casi, Trident non ritenta automaticamente l'operazione in caso di errore. 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.

Eliminare un PV con gli snapshot associati

Quando si elimina un volume persistente con gli snapshot associati, il volume Trident corrispondente viene aggiornato allo "stato di eliminazione". Rimuovere gli snapshot del volume per eliminare il volume 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
```

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

Link correlati

- "Snapshot dei volumi"
- "VolumeSnapshotClass"

Lavora con gli snapshot dei gruppi di volumi di Trident

Snapshot del gruppo di volumi Kubernetes di volumi persistenti (PV). NetApp Trident offre la possibilità di creare snapshot di più volumi (un gruppo di snapshot di volume). Questo snapshot del gruppo di volumi rappresenta copie di più volumi acquisite nello stesso momento.

NOTA VolumeGroupSnapshot è una funzionalità beta di Kubernetes con API beta. La versione minima richiesta per VolumeGroupSnapshot è Kubernetes 1.32.

Creare snapshot del gruppo di volumi

Lo snapshot del gruppo di volumi è supportato con i seguenti driver di archiviazione:

- `ontap-san`driver - solo per i protocolli iSCSI e FC, non per il protocollo NVMe/TCP.
- `ontap-san-economy` - solo per il protocollo iSCSI.
- `ontap-nas`

NOTA Lo snapshot del gruppo di volumi non è supportato per i sistemi di archiviazione NetApp ASA r2 o AFX.

Prima di iniziare

- Assicurati che la versione di Kubernetes sia K8s 1.32 o successiva.
- 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 di Kubernetes non include il controller di snapshot esterno e i CRD, fare riferimento a Implementare un controller per lo snapshot dei volumi .

NOTA	Non creare uno snapshot controller se si creano snapshot di gruppi di volumi su richiesta in un ambiente GKE. GKE utilizza un controller di snapshot integrato e nascosto.

- Nel controller snapshot YAML, imposta `CSIVolumeGroupSnapshot` feature gate su 'true' per garantire che lo snapshot del gruppo di volumi sia abilitato.
- Creare le classi di snapshot del gruppo di volumi richieste prima di creare uno snapshot del gruppo di volumi.
- Assicurarsi che tutti i PVC/volumi siano sullo stesso SVM per poter creare VolumeGroupSnapshot.

Fasi

- Creare una VolumeGroupSnapshotClass prima di creare una VolumeGroupSnapshot. Per ulteriori informazioni, fare riferimento a "Classe VolumeGroupSnapshot".

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

- Creare PVC con le etichette richieste utilizzando le classi di archiviazione esistenti oppure aggiungere queste etichette ai PVC esistenti.
  1. Definisci la chiave e il valore dell'etichetta in base alle tue esigenze

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- Crea un VolumeGroupSnapshot con la stessa etichetta (`consistentGroupSnapshot: groupA`) specificato nel PVC.

  Questo esempio crea uno snapshot del gruppo di volumi:

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

Recupera i dati del volume utilizzando uno snapshot di gruppo

È possibile ripristinare singoli volumi persistenti utilizzando i singoli snapshot creati come parte dello snapshot del gruppo di volumi. Non è possibile ripristinare lo snapshot del gruppo di volumi come unità.

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
```

NOTA	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

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

Per ulteriori informazioni, vedere "Ripristino del volume in-place da uno snapshot".

Elimina un PV con snapshot di gruppo associati

Quando si elimina uno snapshot del volume di gruppo:

- È possibile eliminare i VolumeGroupSnapshot nel loro insieme, non i singoli snapshot del gruppo.
- Se i PersistentVolume vengono eliminati mentre esiste uno snapshot per quel PersistentVolume, Trident sposterà quel volume in uno stato di "eliminazione" perché lo snapshot deve essere rimosso prima che il volume possa essere rimosso in modo sicuro.
- Se è stato creato un clone utilizzando uno snapshot raggruppato e in seguito il gruppo deve essere eliminato, verrà avviata un'operazione di suddivisione su clone e il gruppo non potrà essere eliminato finché la suddivisione non sarà completata.

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-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. Creare il controller di snapshot.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

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

Link correlati

- "Classe VolumeGroupSnapshot"
- "Snapshot dei volumi"

Gestire e monitorare Trident

Upgrade Trident (Aggiorna server)

Upgrade Trident (Aggiorna server)

A partire dalla release 24,02, 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. Vi consigliamo di effettuare l'aggiornamento almeno una volta all'anno per usufruire delle nuove funzioni di Trident.

Considerazioni prima dell'aggiornamento

Quando si effettua l'aggiornamento alla versione più recente di Trident, tenere presente quanto segue:

- Dovrebbe essere installata una sola istanza di Trident in tutti gli spazi dei nomi di un determinato cluster Kubernetes.
- Trident 23,07 e versioni successive richiedono snapshot di volume v1 e non supportano più snapshot alfa o beta.
- Quando si esegue l'aggiornamento, è importante fornire `parameter.fsType` in `StorageClasses` usato da Trident. Puoi 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 Trident seguono una convenzione di denominazione basata sulla data `YY.MM`, dove "YY" è l'ultima cifra dell'anno e "MM" è il mese. I rilasci di DOT seguono una `YY.MM.X` convenzione, 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 24,06 (o qualsiasi versione a 24,06 punti) a 25,06.
- Se si sta eseguendo l'aggiornamento da una release al di fuori della finestra a quattro release, eseguire un aggiornamento in più fasi. Utilizzare le istruzioni di aggiornamento per il "versione precedente" quale si sta eseguendo l'aggiornamento per passare alla versione più recente adatta alla finestra a quattro release. Ad esempio, se si utilizza 23,07 e si desidera eseguire l'aggiornamento a 25,06:
  a. Primo aggiornamento dal 23.07 al 24.06.
  b. Quindi aggiorna dalla versione 24.06 alla versione 25.06.

Fase 2: Determinare il metodo di installazione originale

Per determinare quale versione è stata utilizzata per l'installazione originale di Trident:

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

Fase 3: Selezionare un metodo di aggiornamento

In generale, dovresti eseguire l'aggiornamento utilizzando lo stesso metodo usato per l'installazione iniziale, tuttavia puoi "passare da un metodo di installazione all'altro". Ci sono due opzioni per aggiornare Trident.

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

SUGGERIMENTO	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

Comprendi il flusso di lavoro di aggiornamento dell'operatore Trident

Prima di utilizzare l'operatore Trident per aggiornare Trident, è necessario comprendere i processi in background che si verificano durante l'aggiornamento. 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

Una delle tante "Vantaggi dell'utilizzo dell'operatore Trident" modalità per installare e aggiornare Trident è la gestione automatica degli oggetti Trident e Kubernetes senza interrompere i volumi montati esistenti. In questo modo, Trident può supportare gli aggiornamenti senza tempi di inattività, ovvero "*rolling updates*" offline. 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.

SUGGERIMENTO	Per ulteriori informazioni sull'architettura Trident nel cluster Kubernetes, fare riferimento a "Architettura 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 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 Trident utilizzando l'operatore Trident o Helm

Puoi aggiornare Trident utilizzando l'operatore Trident manualmente o tramite Helm. Puoi aggiornare da un'installazione dell'operatore Trident a un'altra installazione dell'operatore Trident oppure aggiornare da un'installazione `tridentctl` a una versione dell'operatore Trident. Esamina "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 con ambito cluster a un'altra installazione dell'operatore Trident con ambito cluster. Tutte le versioni Trident utilizzano un operatore con ambito cluster.

NOTA	Per eseguire l'aggiornamento da Trident installato utilizzando l'operatore con spazio dei nomi (versioni da 20,07 a 20,10), utilizza le istruzioni di aggiornamento di "versione installata"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 1,25 o versione successiva, 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 Trident:

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

2. Aggiorna il `operator.yaml` , `tridentorchestrator_cr.yaml` , E `post_1_25_bundle.yaml` con il registro e i percorsi immagine per la versione a cui si sta effettuando l'aggiornamento (ad esempio 25.06) e il segreto corretto.
3. Eliminare l'operatore Trident utilizzato per installare l'istanza Trident corrente. Ad esempio, se si esegue l'aggiornamento dalla versione 25.02, eseguire il seguente comando:

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. 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.
5. Installa Trident utilizzando il file YAML del bundle corretto per il tuo ambiente, dove *<bundle.yaml>* è `bundle_pre_1_25.yaml` O `bundle_post_1_25.yaml` in base alla versione di Kubernetes. Ad esempio, se si installa Trident 25.06.0, eseguire il seguente comando:

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. Modifica la torcia del tridente per includere l'immagine 25.06.0.

Aggiornare un'installazione Helm

È possibile aggiornare un'installazione di Trident Helm.

ATTENZIONE	Quando si aggiorna un cluster Kubernetes da 1,24 a 1,25 o versione successiva su `true` cui è installato Trident, è necessario aggiornare Values.yaml per impostarlo `excludePodSecurityPolicy` o aggiungerlo `--set excludePodSecurityPolicy=true` al `helm upgrade` comando prima di poter aggiornare il cluster.

Se hai già aggiornato il tuo cluster Kubernetes dalla 1,24 alla 1,25 senza aggiornare il timone Trident, l'aggiornamento del timone non riuscirà. Per eseguire l'aggiornamento del timone, eseguire questi passaggi come prerequisiti:

1. Installare il plugin helm-mapkubeapis da https://github.com/helm/helm-mapkubeapis.
2. Eseguire un ciclo di asciugatura per la release Trident nello spazio dei nomi in cui è installato Trident. In questo modo vengono elencate le risorse che verranno ripulite.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. Eseguire una corsa completa con il timone per eseguire la pulizia.

```
helm mapkubeapis trident  --namespace trident
```

Fasi

1. Se "Installato Trident utilizzando Helm", puoi usare `helm upgrade trident netapp-trident/trident-operator --version 100.2610.0` per aggiornare in un unico passaggio. Se non hai aggiunto il Helm repo o non puoi usarlo per aggiornare:
   a. Scaricare la versione più recente di Trident dal sito "La sezione *Assets* su GitHub".
   b. Usa il `helm upgrade` comando dove `trident-operator-26.10.1.tgz` riflette la versione a cui vuoi eseguire l'aggiornamento.

```
helm upgrade <name> trident-operator-26.10.1.tgz
```

> **NOTA** 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.

> **NOTA** Prima di passare da un metodo di installazione all'altro, rivedere "Passaggio da un metodo di installazione all'altro".

Fasi

1. Scarica la versione più recente di Trident.

```
# Download the release required [26.10.1]
mkdir 26.10.1
cd 26.10.1
wget
https://github.com/NetApp/trident/releases/download/v26.10.0/trident-
installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.tar.gz
cd trident-installer
```

2. Creare il `tridentorchestrator` CRD dal manifesto.

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

3. Implementare l'operatore 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 una `TridentOrchestrator` CR per l'installazione di 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:                v26.10.0
```

Aggiorna Trident con tridentctl

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

A proposito di questa attività

La disinstallazione e la reinstallazione di Trident funge da aggiornamento. Quando si disinstalla Trident, il PVC (Persistent Volume Claim) e il PV (Persistent Volume Claim) utilizzati dall'implementazione Trident non vengono eliminati. I PVC che sono già stati sottoposti a provisioning rimarranno disponibili mentre Trident è offline, e Trident eseguirà il provisioning dei volumi per qualsiasi PVC creato nel frattempo dopo il ritorno online.

Prima di iniziare

Rivedi "Selezionare un metodo di aggiornamento" prima di aggiornare usando `tridentctl`.

Fasi

1. Eseguire il comando di disinstallazione in `tridentctl` per rimuovere tutte le risorse associate a Trident, ad eccezione dei CRD e degli oggetti correlati.

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

2. Reinstallare Trident. Fare riferimento a "Installare Trident usando tridentctl".

IMPORTANTE	Non interrompere il processo di aggiornamento. Assicurarsi che il programma di installazione venga completato.

Gestisci Trident usando tridentctl

```
https://github.com/NetApp/trident/releases["Pacchetto di installazione
Trident"^]Include l' `tridentctl`utilità della riga di comando per fornire
un semplice accesso a Trident. Gli utenti Kubernetes con Privileges
sufficiente possono usarlo per installare Trident o gestire il namespace
che contiene l'pod 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]
```

L'utilità Trident `tridentctl` supporta i seguenti comandi e flag globali.

▼ **Comandi**

`create`

Aggiungere una risorsa a Trident.

delete
Rimuovere una o più risorse da Trident.

get
Ottieni una o più risorse da Trident.

help
Aiuto su qualsiasi comando.

images
Stampare una tabella delle immagini contenitore richieste da Trident.

import
Importare una risorsa esistente in Trident.

install
Installare Trident.

logs
Stampare i registri da Trident.

send
Inviare una risorsa da Trident.

uninstall
Disinstallare Trident.

update
Modificare una risorsa in Trident.

update backend state
Sospendere temporaneamente le operazioni di backend.

upgrade
Aggiornare una risorsa in Trident.

version
Stampare la versione di 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.

NOTA	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 delle implementazioni Trident.

`-o, --output string`
Formato di output. Uno tra json|yaml|name|wide|ps (impostazione predefinita).

`-s, --server string`
Indirizzo/porta dell'interfaccia REST Trident.

ATTENZIONE	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 il `create` comando per aggiungere una risorsa a Trident.

```
tridentctl create [option]
```

Opzioni

`backend`: Aggiungere un backend a Trident.

eliminare

Utilizzare il `delete` comando per rimuovere una o più risorse da Trident.

```
tridentctl delete [option]
```

Opzioni

`backend`: Eliminare uno o più backend di archiviazione da Trident.
`snapshot`: Eliminare uno o più snapshot di volume da Trident.
`storageclass`: Eliminare una o più classi di archiviazione da Trident.
`volume`: Eliminare uno o più volumi di archiviazione da Trident.

ottieni

Utilizzare il `get` comando per ottenere una o più risorse da Trident.

```
tridentctl get [option]
```

Opzioni

`backend`: Ottenere uno o più backend di archiviazione da Trident.
`snapshot`: Ottenere uno o più snapshot da Trident.
`storageclass`: Ottenere una o più classi di archiviazione da Trident.
`volume`: Ottenere uno o più volumi da 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` i flag per stampare una tabella delle immagini contenitore richieste da Trident.

```
tridentctl images [flags]
```

Allarmi

`-h`, `--help`: Guida per le immagini.
`-v`, `--k8s-version string`: Versione semantica del cluster Kubernetes.

importa volume

Utilizzare il `import volume` comando per importare un volume esistente in 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 i `install` flag per installare Trident.

```
tridentctl install [flags]
```

Allarmi

`--autosupport-image string`: L'immagine del contenitore per Autosupport Telemetry (predefinita "netapp/trident autosupport:<current-version>").
`--autosupport-proxy string` : Indirizzo/porta di un proxy per l'invio di dati di telemetria di Autosupport.
`--enable-node-prep` : Tentativo di installare i pacchetti richiesti sui nodi.
`--generate-custom-yaml` : Genera file YAML senza installare nulla.
`-h` , `--help` : Aiuto per l'installazione.
`--http-request-timeout` : Sostituisci il timeout della richiesta HTTP per l'API REST del controller Trident (predefinito 1m30s).
`--image-registry string` : L'indirizzo/porta di un registro di immagini interno.
`--k8s-timeout duration` : Timeout per tutte le operazioni Kubernetes (predefinito 3m0s).
`--kubelet-dir string` : Posizione host dello stato interno di kubelet (predefinito "/var/lib/kubelet").
`--log-format string` : Formato di registrazione Trident (testo, json) (predefinito "testo").
`--node-prep` : consente a Trident di preparare i nodi del cluster Kubernetes per gestire i volumi utilizzando il protocollo di archiviazione dati specificato. **Attualmente, `iscsi` è l'unico valore supportato.**

A partire da OpenShift 4.19, la versione minima Trident supportata per questa funzionalità è 25.06.1.
`--pv string` : Il nome del PV legacy utilizzato da Trident, assicura che non esista (predefinito "trident").
`--pvc string` : Il nome del PVC legacy utilizzato da Trident, assicura che questo non esista (predefinito "trident").
`--silence-autosupport` : Non inviare automaticamente i bundle di supporto automatico a NetApp (valore predefinito: true).
`--silent` : Disabilita la maggior parte degli output durante l'installazione.
`--trident-image string` : L'immagine Trident da installare.
`--k8s-api-qps` : Limite di query al secondo (QPS) per le richieste API di Kubernetes (predefinito 100; facoltativo).
`--use-custom-yaml` : Utilizzare tutti i file YAML esistenti nella directory di installazione.
`--use-ipv6` : Utilizza IPv6 per la comunicazione di Trident.

registri

Utilizzare `logs` i flag per stampare i registri da Trident.

```
tridentctl logs [flags]
```

Allarmi

`-a`, `--archive`: Creare un archivio di supporto con tutti i registri, se non diversamente specificato.
`-h`, `--help`: Guida per i registri.
`-l`, `--log string`: Registro Trident da visualizzare. Uno di 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`: Ottiene i log per l'istanza contenitore precedente, se esiste.
`--sidecars`: Ottenere i tronchi per i contenitori del sidecar.

invia

Utilizzare il `send` comando per inviare una risorsa da Trident.

```
tridentctl send [option]
```

Opzioni

`autosupport`: Inviare un archivio AutoSupport a NetApp.

disinstallazione

Utilizzare `uninstall` i flag per disinstallare Trident.

```
tridentctl uninstall [flags]
```

Allarmi

`-h, --help`: Guida per la disinstallazione.
`--silent`: Disattivare la maggior parte dell'output durante la disinstallazione.

aggiornamento

Utilizzare il `update` comando per modificare una risorsa in Trident.

```
tridentctl update [option]
```

Opzioni

`backend`: Aggiornare un backend in Trident.

aggiorna stato backend

Utilizzare `update backend state` comando per sospendere o riprendere le operazioni di backend.

```
tridentctl update backend state <backend-name> [flag]
```

Punti da considerare

- Se un backend viene creato utilizzando un TridentBackendConfig (tbc), non è possibile aggiornare il backend utilizzando un `backend.json` file.
- Se il `userState` è stato impostato in un tbc, non può essere modificato utilizzando il `tridentctl update backend state <backend-name> --user-state suspended/normal` comando .
- Per recuperare la capacità di impostare il `userState` tridentctl via dopo che è stato impostato tramite tbc, il `userState` campo deve essere rimosso dal tbc. Questo può essere fatto usando il `kubectl edit tbc` comando. Una volta rimosso il `userState` campo, è possibile utilizzare il `tridentctl update backend state` comando per modificare il `userState` di un backend.
- Utilizzare il `tridentctl update backend state` per modificare il `userState`. È anche possibile aggiornare il `userState` file Using `TridentBackendConfig` o `backend.json` ; questo attiva una reinizializzazione completa del backend e può richiedere molto tempo.

  #### 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` e `Import Volume` sono in pausa.
- `CloneVolume`, , `ResizeVolume`, , `PublishVolume` `UnPublishVolume`, , `CreateSnapshot` `GetSnapshot` `RestoreSnapshot`, , , , `DeleteSnapshot` `RemoveVolume`, , `GetVolumeExternal` `ReconcileNodeAccess` rimangono disponibili.

È inoltre possibile aggiornare lo stato backend utilizzando il `userState` campo nel file di configurazione backend `TridentBackendConfig` o `backend.json`. Per ulteriori informazioni, fare riferimento a "Opzioni per la gestione dei backend" e "Eseguire la gestione del back-end con kubectl".

Esempio:

JSON

Per aggiornare utilizzando il file, procedere come segue `userState` `backend.json` :

1. Modificare il `backend.json` file per includere il `userState` campo con il valore impostato su 'sospeso'.
2. Aggiorna il backend utilizzando `tridentctl update backend` comando e il percorso per l'aggiornamento `backend.json` file.

   Esempio: `tridentctl update backend -f /<path to backend JSON file>/backend.json -n trident`

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "<redacted>",
  "svm": "nas-svm",
  "backendName": "customBackend",
  "username": "<redacted>",
  "password": "<redacted>",
  "userState": "suspended"
}
```

YAML

È possibile modificare il tbc dopo averlo applicato utilizzando il `kubectl edit <tbc-name> -n <namespace>` comando . Nell'esempio riportato di seguito viene aggiornato lo stato backend per la sospensione mediante l' `userState: suspended` opzione:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  backendName: customBackend
  storageDriverName: ontap-nas
  managementLIF: <redacted>
  svm: nas-svm
  userState: suspended
  credentials:
    name: backend-tbc-ontap-nas-secret
```

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.

Supporto plugin

Tridentctl supporta plugin simili a kubectl. Tridentctl rileva un plugin se il nome del file binario del plugin segue lo schema "tridentctl-<plugin>", e il binario si trova in una cartella elencata nella variabile di ambiente PATH. Tutti i plugin rilevati sono elencati nella sezione dei plugin della guida tridentctl. In alternativa, è possibile

limitare la ricerca specificando una cartella di plugin nella variabile Environment TRIDENTCTL_PLUGIN_PATH (esempio: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). Se si utilizza la variabile, tridenctl ricerca solo nella cartella specificata.

Monitora le operazioni di storage di Trident

Trident fornisce un set di endpoint di misurazione Prometheus che è possibile utilizzare per monitorare le prestazioni Trident.

Panoramica

Le metriche fornite da Trident consentono di:

- Tenere sotto controllo lo stato di salute e la configurazione di 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. È possibile esaminare il tempo necessario a Trident per comunicare con i backend ed eseguire le operazioni.

NOTA	Per impostazione predefinita, le metriche di Trident sono esposte sulla porta di destinazione `8001` al `/metrics` punto finale. Queste metriche sono **abilitate per impostazione predefinita** quando Trident è installato. È possibile configurare l'utilizzo delle metriche Trident tramite HTTPS sulla porta `8444` anche.

Di cosa hai bisogno

- Un cluster Kubernetes con 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

Dovresti definire un target Prometheus per raccogliere le metriche e ottenere informazioni sui backend gestiti Trident , sui volumi che crea e così via. Vedere"Documentazione dell'operatore Prometheus" .

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` servizio e cerca specificamente il `metrics` endpoint del servizio. Di conseguenza, Prometheus è ora configurato per comprendere le metriche di Trident.

Oltre alle metriche disponibili direttamente da Trident, kubelet espone molte `kubelet_volume_*` metriche tramite il proprio endpoint di misurazione. Kubelet può fornire informazioni sui volumi collegati, sui pod e sulle altre operazioni interne gestite. Fare riferimento alla "qui".

Utilizza le metriche Trident tramite HTTPS

Per utilizzare le metriche Trident tramite HTTPS (porta 8444), è necessario modificare la definizione di ServiceMonitor per includere la configurazione TLS. Devi anche copiare il `trident-csi` segreto dal `trident` namespace allo spazio dei nomi in cui è in esecuzione Prometheus. Puoi farlo usando il seguente comando:

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

Un esempio di metriche ServiceMonitor per HTTPS si presenta così:

```
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:
    - interval: 15s
```

```
    path: /metrics
    port: https-metrics
    scheme: https
    tlsConfig:
      ca:
        secret:
          key: caCert
          name: trident-csi
      cert:
        secret:
          key: clientCert
          name: trident-csi
      keySecret:
        key: clientKey
        name: trident-csi
      serverName: trident-csi
```

Trident supporta le metriche HTTPS in tutti i metodi di installazione: tridentctl, Helm chart e Operator:

- Se stai utilizzando il `tridentctl install` comando, puoi passare il `--https-metrics` flag per abilitare le metriche HTTPS.
- Se si utilizza il grafico Helm, è possibile impostare `httpsMetrics` parametro per abilitare le metriche HTTPS.
- Se si utilizzano file YAML, è possibile aggiungere `--https_metrics` bandiera al `trident-main` contenitore nel `trident-deployment.yaml` file.

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 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 A RIPOSO da 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 Trident**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Ottenere informazioni sull'utilizzo di 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

NOTA 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
```

Ulteriori informazioni sulla telemetria di Trident AutoSupport

Per impostazione predefinita, Trident invia quotidianamente le metriche Prometheus e le informazioni di base di backend a NetApp.

- Per impedire a Trident di inviare metriche Prometheus e informazioni di base di backend a NetApp, passare il `--silence-autosupport` flag durante l'installazione di Trident.
- Trident può inoltre inviare i log dei container al supporto NetApp on-demand tramite `tridentctl send autosupport`. Sarà necessario attivare Trident per caricare i suoi registri. Prima di inviare i log, è necessario accettare NetApp "direttiva sulla privacy".
- Se non specificato, Trident recupera i registri dalle ultime 24 ore.
- È possibile specificare il periodo di conservazione del registro con il `--since` flag. Ad esempio: `tridentctl send autosupport --since=1h`. Queste informazioni vengono raccolte e inviate tramite un `trident-autosupport` contenitore installato insieme a Trident. È possibile ottenere l'immagine contenitore in "Trident AutoSupport".
- Trident AutoSupport non raccoglie né trasmette Personally Identifiable Information (PII) o Personal Information. Viene fornito con un "EULA" che non è applicabile all'immagine del container Trident stessa. Puoi saperne di più sull'impegno di NetApp per la sicurezza e la fiducia dei dati "qui".

Un esempio di payload inviato da Trident è simile al seguente:

```
---
items:
  - backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
    protocol: file
    config:
      version: 1
      storageDriverName: ontap-nas
      debug: false
      debugTraceFlags: null
      disableDelete: false
      serialNumbers:
        - nwkvzfanek_SN
      limitVolumeSize: ""
    state: online
    online: true
```

- I messaggi AutoSupport vengono inviati all'endpoint AutoSupport di NetApp. Se si utilizza un registro privato per memorizzare le immagini container, è possibile utilizzare `--image-registry` allarme.
- È inoltre possibile configurare gli URL proxy generando i file YAML di installazione. Per eseguire questa operazione, utilizzare `tridentctl install --generate-custom-yaml` Per creare i file YAML e aggiungere `--proxy-url` argomento per `trident-autosupport` container in `trident-deployment.yaml`.

Disattiva metriche 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 Trident

Utilizzare lo stesso metodo per disinstallare Trident utilizzato per installare Trident.

A proposito di questa attività

- 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 Trident e reinstallare la versione precedente utilizzando le istruzioni specifiche per tale aggiornamento"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 Trident non rimuove i CRD o gli oggetti correlati creati da Trident. Se è necessario rimuovere completamente Trident e tutti i relativi dati, fare riferimento alla sezione "Rimuovere completamente Trident e CRD".

Prima di iniziare

Se stai decommissionando i cluster Kubernetes, devi eliminare tutte le applicazioni che utilizzano i volumi creati da 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 Trident utilizzato per installarlo. Prima di disinstallare, verificare quale versione è stata utilizzata per installare Trident in origine.

1. Utilizzare `kubectl get pods -n trident` esaminare i pod.
   - Se non è presente alcun pannello operatore, Trident è stato installato utilizzando `tridentctl`.
   - Se è presente un quadro di comando, Trident è stato installato utilizzando l'operatore Trident manualmente o utilizzando Helm.
2. Se è presente un pannello operatore, utilizzare `kubectl describe tproc trident` per determinare se Trident è stato installato utilizzando Helm.
   - Se è presente un'etichetta Helm, Trident è stato installato utilizzando Helm.
   - Se non è presente alcuna etichetta Helm, 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 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 il `TridentOrchestrator` CR utilizzato per distribuire Trident, si istruisce l'operatore a disinstallare Trident. L'operatore elabora la rimozione `TridentOrchestrator` e procede alla rimozione della distribuzione Trident e del daemonset, eliminando i pod Trident creati durante l'installazione.

   ```
   kubectl delete -f deploy/<bundle.yaml> -n <namespace>
   ```

Disinstallare l'installazione di Helm

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

```
#List the Helm release corresponding to the 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 il `uninstall` comando in `tridentctl` per rimuovere tutte le risorse associate a Trident, ad eccezione dei CRD e degli oggetti correlati:

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

Trident per Docker

Prerequisiti per l'implementazione

È necessario installare e configurare i prerequisiti del protocollo necessari sull'host prima di poter distribuire 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
```

ATTENZIONE	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
```

NOTA 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
```

NOTA 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.

NOTA

- 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
```

Strumenti FC

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

- Quando si utilizzano nodi di lavoro che eseguono RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) con FC PVS, specificare il `discard` mount Option in StorageClass per eseguire il recupero dello spazio in linea. Fare riferimento alla "Documentazione di Red Hat".

RHEL 8+

1. Installare i seguenti pacchetti di sistema:

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. Abilitare il multipathing:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

NOTA Assicurarsi `etc/multipath.conf` contiene `find_multipaths no` sotto `defaults`.

3. Assicurarsi che `multipathd` sia in esecuzione:

```
sudo systemctl enable --now multipathd
```

Ubuntu

1. Installare i seguenti pacchetti di sistema:

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. 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
```

NOTA Assicurarsi `etc/multipath.conf` contiene `find_multipaths no` sotto

```
defaults.
```

3. Assicurarsi che `multipath-tools` sia attivato e in esecuzione:

```
sudo systemctl status multipath-tools
```

Implementa Trident

Trident per Docker offre un'integrazione diretta con l'ecosistema Docker per le piattaforme storage NetApp. Supporta il provisioning e la gestione delle risorse di storage dalla piattaforma di storage agli host Docker, con un framework per aggiungere altre piattaforme in futuro.

È possibile eseguire più istanze di Trident 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

Consultare la "prerequisiti per l'implementazione". Una volta soddisfatti i prerequisiti, è possibile distribuire Trident.

Metodo del plugin gestito da Docker (versione 1.13/17.03 e successive)

NOTA	**Prima di iniziare** Se è stato utilizzato Trident pre Docker 1,13/17,03 nel metodo daemon tradizionale, prima di utilizzare il metodo plugin gestito, è necessario arrestare il processo Trident e riavviare il daemon Docker.

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 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 Trident per utilizzare 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:

```
----trident-installer-26.10.1.tar.gz
wget https://github.com/NetApp/trident/releases/download/10.0/
tar zxf trident-installer-26.10.1.tar.gz
```

```
.. Move to a location in the bin path:
+
[source,console]
```

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

```
.. Create a location for the configuration file:
+
```

```
[source,console]
```

sudo mkdir -p /etc/netappdvp

```
.. Create the configuration file:
+
[source,console]
```

cat << EOF > /etc/netappdvp/ontap-nas.json

```
+
[source,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

```
. After placing the binary and creating the configuration file, start the
Trident daemon using the desired configuration file.
+
[source,console]
```

sudo trident --config=/etc/netappdvp/ontap-nas.json

```
+
NOTE: Unless specified, the default name for the volume driver is
"netapp".
+
After the daemon is started, you can create and manage volumes by using
the Docker CLI interface.

. Create a volume:
+
[source,console]
```

docker volume create -d netapp --name trident_1

```
. Provision a Docker volume when starting a container:
+
[source,console]
```

docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol alpine ash

```
. Remove a Docker volume:
+
[source,console]
```

docker volume rm trident_1

```
+
[source,console]
```

docker volume rm trident_2

```
== Start Trident at system startup

A sample unit file for systemd based systems can be found at
`contrib/trident.service.example` in the Git repo. To use the file with
RHEL, do the following:

. Copy the file to the correct location.
+
You should use unique names for the unit files if you have more than one
instance running.
+
[source,console]
```

cp contrib/trident.service.example /usr/lib/systemd/system/trident.service

```
. Edit the file, change the description (line 2) to match the driver name
and the configuration file path (line 9) to reflect your environment.
. Reload systemd for it to ingest changes:
+
[source,console]
```

systemctl daemon-reload

```
. Enable the service.
+
This name varies depending on what you named the file in the
`/usr/lib/systemd/system` directory.
+
[source,console]
```

systemctl enable trident

```
. Start the service.
+
[source,console]
```

systemctl start trident

```
. View the status.
+
[source,console]
```

systemctl status trident

```
NOTE: Any time you modify the unit file, run the `systemctl daemon-reload`
command for it to be aware of the changes.
```

Aggiornare o disinstallare Trident

Puoi eseguire l'upgrade sicuro di 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 saranno in grado di montare i volumi fino a quando il plugin non sarà nuovamente in esecuzione. Nella maggior parte dei casi, si tratta di pochi secondi.

Eseguire l'upgrade

Eseguire i passaggi riportati di seguito per eseguire l'upgrade di Trident per Docker.

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
```

NOTA La versione 18,01 di Trident sostituisce il nDVP. È necessario eseguire l'aggiornamento direttamente dall' `netapp/ndvp-plugin`immagine all' `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
```

IMPORTANTE Se si sta eseguendo l'aggiornamento da una vecchia versione di Trident (precedente alla 20,10) a Trident 20,10 o versione successiva, potrebbe verificarsi un errore. Per ulteriori informazioni, fare riferimento a "Problemi noti". Se si verifica l'errore, si dovrebbe prima disabilitare il plugin, quindi rimuovere il plugin, e quindi installare la versione Trident richiesta passando un parametro di configurazione extra: `docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all-permissions config=config.json`

Disinstallare

Per disinstallare 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

Puoi creare, clonare e rimuovere facilmente volumi utilizzando comandi standard `docker volume` con il nome del driver Trident specificato quando necessario.

Creare un volume

- Creare un volume con un driver utilizzando il nome predefinito:

```
docker volume create -d netapp --name firstVolume
```

- Creazione di un volume con un'istanza Trident specifica:

```
docker volume create -d ntap_bronze --name bronzeVolume
```

NOTA Se non si specifica alcuna "opzioni", vengono utilizzate le impostazioni predefinite del driver.

- Sostituisci la dimensione predefinita del volume. Per creare un volume da 20 GiB con un driver, vedi l'esempio seguente:

```
docker volume create -d netapp --name my_vol --opt size=20G
```

SUGGERIMENTO 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
```

IMPORTANTE	Quando si utilizza `solidfire-san` driver, l'esempio precedente elimina e cancella il volume.

Eseguire i passaggi riportati di seguito per eseguire l'upgrade di Trident per Docker.

Clonare un volume

Quando si utilizza il `ontap-nas` , `ontap-san` , E `solidfire-san` driver di archiviazione, Trident può clonare i volumi. Quando si utilizza il `ontap-nas-flexgroup` O `ontap-nas-economy` driver, la clonazione non è supportata. La creazione di un nuovo volume da un volume esistente comporterà 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) utilizzando i contenitori Trident **solo** se non hanno partizioni e se il loro filesystem è supportato da Trident (ad esempio: Un file `ext4` formattato `/dev/sdc1` non sarà accessibile tramite 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, iSCSI e FC includono quanto segue:

Opzione	Descrizione
`size`	La dimensione predefinita del volume è 1 GiB.

Opzione	Descrizione
`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 creati automaticamente istantanee 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 al giorno e due snapshot settimanali. I dati conservati in uno snapshot possono essere recuperati navigando nella `.snapshot` directory di 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.
`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 abilitato sul backend, qualsiasi volume sottoposto a provisioning in Trident sarà abilitato NAE. Per ulteriori informazioni, fare riferimento a: "Come funziona 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:

- Crea un volume da 10 GiB:

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- Crea un volume da 100 GiB 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
```

La dimensione minima del volume è 20 MiB.

Se la riserva istantanea non viene specificata e il criterio snapshot è `none`, Trident utilizza una riserva istantanea del 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 un criterio snapshot e accettare la riserva snapshot predefinita di ONTAP (in genere 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, predefinita è 1 GiB o voce di configurazione … "defaults": {"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 Trident utilizzando il metodo del plugin gestito consigliato (ad esempio, utilizzando `docker plugin` i comandi), 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.

Gestione di più istanze di 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 tue configurazioni Trident.

Opzioni di configurazione globale

Queste opzioni di configurazione sono valide per tutte le configurazioni di Trident, a prescindere dalla piattaforma di storage utilizzata.

Opzione	Descrizione	Esempio
`version`	Numero di versione del file di configurazione	`1`

Opzione	Descrizione	Esempio
`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`

SUGGERIMENTO

Non utilizzare `storagePrefix` (incluso il valore predefinito) per i backend dell'elemento. Per impostazione predefinita, il `solidfire-san` driver ignora questa impostazione e non utilizza un prefisso. NetApp consiglia di utilizzare un ID tenant specifico per la mappatura dei volumi di Docker o i dati degli attributi popolati con la versione di Docker, le informazioni dei driver e il nome raw di Docker nei casi in cui sia stata utilizzata la mappatura 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`

Opzione	Descrizione	Esempio
`dataLIF`	Indirizzo IP del protocollo LIF. **Driver NAS ONTAP**: NetApp consiglia di specificare `dataLIF`. Se non viene fornita, Trident recupera le LIF dati dalla SVM. È possibile specificare un nome di dominio completo (FQDN) da utilizzare per le operazioni di montaggio NFS, consentendo di creare un DNS round-robin per bilanciare il carico su più LIF dati. **Driver SAN ONTAP**: Non specificare per iSCSI o FC. Trident utilizza "Mappa LUN selettiva ONTAP" per rilevare le LIF iSCSI o FC necessarie per stabilire una sessione multi-path. Viene generato un avviso se `dataLIF` è definito esplicitamente.	`10.0.0.2`
`svm`	Macchina virtuale per lo storage da utilizzare (obbligatorio, se la LIF di gestione è una LIF del cluster)	`svm_nfs`
`username`	Nome utente per la connessione al dispositivo di storage	`vsadmin`
`password`	Password per la connessione al dispositivo di storage	`secret`
`aggregate`	Aggregato per il provisioning (facoltativo; se impostato, deve essere assegnato alla SVM). Per il `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 granulare delle opzioni di montaggio NFS; il valore predefinito è "-o nfsfvers=3". **Disponibile solo per `ontap-nas` i driver e `ontap-nas-economy`.** "Fare clic qui per informazioni sulla configurazione degli host NFS".	`-o nfsvers=4`

Opzione	Descrizione	Esempio
`igroupName`	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`	Dimensioni massime del volume richiudibile.	`300g`
`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 solo per `ontap-san` il driver.** Utilizzare per selezionare `iscsi` iSCSI, `nvme` NVMe/TCP o `fcp` SCSI over Fibre Channel (FC).	`iscsi` se vuoto
`limitVolumePoolSize`	**Supportato `ontap-san-economy` `ontap-san-economy` solo per i driver e.** Limita le dimensioni degli FlexVol in driver ONTAP ONTAP-nas-Economy e ONTAP-SAN-Economy.	`300g`

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`	Percentuale di riserva istantanea, 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`

Opzione	Descrizione	Esempio
`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 abilitato sul backend, qualsiasi volume sottoposto a provisioning in Trident sarà abilitato NAE. Per ulteriori informazioni, fare riferimento a: "Come funziona Trident con NVE e NAE".	vero
`unixPermissions`	Opzione NAS per i volumi NFS con provisioning, per impostazione predefinita su `777`	`777`
`snapshotDir`	Opzione NAS per l'accesso alla `.snapshot` directory.	Consente o disabilita l'accesso alla `.snapshot` directory. Valori validi: `true`, `false`. Il comportamento predefinito potrebbe variare in base alla versione di ONTAP e Trident. Impostare esplicitamente per garantire un comportamento coerente.
`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`	Criterio di tiering da utilizzare, il valore predefinito è `none`.	`none`
`skipRecoveryQueue`	Durante l'eliminazione del volume, ignorare la coda di ripristino nell'archiviazione ed eliminare immediatamente il volume.	``

Opzioni di scalabilità

`ontap-nas`E `ontap-san` crea una ONTAP FlexVol per ogni volume di Docker. ONTAP supporta fino a 1000 FlexVol per nodo del cluster con un massimo di 12.000 FlexVol Volumes. Se i requisiti del tuo volume Docker soddisfano tali requisiti, il driver è la soluzione NAS preferita, `ontap-nas` a causa delle funzionalità aggiuntive offerte da FlexVol, come snapshot Docker-volume-granulari e cloning.

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.

```
`ontap-nas-economy`Il driver crea volumi Docker come qtree ONTAP
all'interno di un pool di volumi 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` driver non supporta le snapshot o il cloning granulari
del volume di Docker.
```

NOTA Al momento il `ontap-nas-economy` driver non è supportato da Docker Swarm, poiché Docker Swarm non orchestra la creazione di volumi su nodi multipli.

```
`ontap-san-economy`Il driver crea volumi Docker come LUN ONTAP all'interno
di un pool condiviso di volumi 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 il `ontap-nas-flexgroup` driver per aumentare il parallelismo a un singolo volume che può raggiungere l'intervallo di petabyte con miliardi di file. Alcuni casi di utilizzo ideali per FlexGroups includono ai/ML/DL, big data e analytics, build software, streaming, repository di file e così via. Trident utilizza tutti gli aggregati assegnati a una SVM durante il provisioning di un volume FlexGroup. Il supporto di FlexGroup in Trident ha anche le seguenti considerazioni:

- Richiede ONTAP versione 9.2 o successiva.
- Al momento della stesura del presente documento, FlexGroups supporta solo NFS v3.
- Si consiglia di attivare gli identificatori NFSv3 a 64 bit per SVM.
- La dimensione minima consigliata per il membro/volume FlexGroup è 100 GiB.
- Il cloning non è supportato per i volumi FlexGroup.

Per informazioni sui FlexGroups e sui carichi di lavoro appropriati per i FlexGroups fare riferimento a "Guida alle Best practice e all'implementazione del volume NetApp FlexGroup".

Per ottenere funzionalità avanzate e su larga scala nello stesso ambiente, è possibile eseguire più istanze di Docker Volume Plugin, con una che utilizza e un'altra `ontap-nas-economy` che utilizza `ontap-nas` .

Ruolo ONTAP personalizzato per Trident

Puoi creare un ruolo cluster ONTAP con Minimum Privileges in modo da non dover utilizzare il ruolo di amministratore ONTAP per eseguire le operazioni in Trident. Quando si include il nome utente in una configurazione backend Trident, Trident utilizza il ruolo del cluster ONTAP creato per eseguire le operazioni.

Per ulteriori informazioni sulla creazione di ruoli personalizzati di Trident, fare riferimento a."Generatore di ruoli

personalizzati Trident"

Utilizzo della CLI di ONTAP

1. Creare un nuovo ruolo utilizzando il seguente comando:

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. Creare un nome utente per l'utente Trident:

```
security login create -username <user_name\> -application ontapi
-authmethod password -role <name_of_role_in_step_1\> -vserver <svm_name\>
-comment "user_description"
security login create -username <user_name\> -application http -authmethod
password -role <name_of_role_in_step_1\> -vserver <svm_name\> -comment
"user_description"
```

3. Associare il ruolo all'utente:

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Utilizzo di System Manager

In Gestione sistema di ONTAP, eseguire le seguenti operazioni:

1. **Crea un ruolo personalizzato**:
   a. Per creare un ruolo personalizzato a livello di cluster, selezionare **Cluster > Impostazioni**.

      (Oppure) per creare un ruolo personalizzato a livello di SVM, selezionare **Storage > Storage VM `required SVM` > > Impostazioni > utenti e ruoli**.

   b. Selezionare l'icona a freccia (→) accanto a **utenti e ruoli**.
   c. Selezionare **+Aggiungi** in **ruoli**.
   d. Definire le regole per il ruolo e fare clic su **Salva**.
2. **Associare il ruolo all'utente Trident**: + eseguire i seguenti passaggi nella pagina **utenti e ruoli**:
   a. Selezionare icona Aggiungi **+** in **utenti**.
   b. Selezionare il nome utente richiesto e scegliere un ruolo nel menu a discesa **ruolo**.
   c. Fare clic su **Save** (Salva).

Per ulteriori informazioni, fare riferimento alle pagine seguenti:

- "Ruoli personalizzati per l'amministrazione di ONTAP" o. "Definire ruoli personalizzati"
- "Lavorare con ruoli e utenti"

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
}
```

▼ **Esempio di SCSI su FC per il driver </code> <code> ONTAP**

```
{
  "version": 1,
  "backendName": "ontap-san-backend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "sanType": "fcp",
  "svm": "trident_svm",
  "username": "vsadmin",
  "password": "password",
  "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
`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

Trova informazioni su problemi e limitazioni noti quando utilizzi 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.

NOTA	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 presenta comportamenti che impediscono a 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 di volumi (incluso Trident) devono essere eseguiti in maniera indipendente su ogni nodo di un cluster Swarm. A causa del funzionamento di ONTAP e del `ontap-nas` funzionamento di e `ontap-san` driver, questi sono gli unici ad essere in grado di operare entro queste limitazioni.

Gli altri piloti sono soggetti a problemi come le condizioni di gara che possono portare alla creazione di un gran numero di volumi per una singola richiesta senza un "vincitore" chiaro; ad esempio, Element ha una caratteristica 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

Raccomandazioni per l'implementazione di Trident

Durante la distribuzione di Trident, utilizza i consigli elencati di seguito.

Eseguire l'implementazione in uno spazio dei nomi dedicato

"Spazi dei nomi" separazione amministrativa tra diverse applicazioni e costituisce un ostacolo alla condivisione delle risorse. Ad esempio, un PVC di uno spazio dei nomi non può essere utilizzato da un altro. Trident fornisce risorse PV a tutti i namespace nel cluster Kubernetes e sfrutta di conseguenza un account di servizio che ha elevato il Privileges.

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".

Procedure consigliate per la configurazione dello storage di Trident

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 fino a 2000 FlexVol Volumes. Il fatto che il numero massimo di volumi sia impostato su 1250 appare molto ragionevole. Tuttavia, se alla SVM viene assegnato solo un nodo oppure se "aggregati" gli aggregati assegnati da un nodo non sono compatibili con il provisioning (ad esempio a causa della capacità), l'altro nodo diventa la destinazione per tutti i volumi con provisioning Trident. Ciò significa che è possibile raggiungere il limite del volume per quel nodo prima che venga raggiunto il `max-volumes` valore, con conseguente impatto sulle operazioni Trident e sugli 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.**

Clonare un volume

NetApp Trident supporta la clonazione dei volumi quando si utilizza `ontap-nas` , `ontap-san` , E `solidfire-san` driver di archiviazione. Quando si utilizza il `ontap-nas-flexgroup` O `ontap-nas-economy` driver, la clonazione non è supportata. La creazione di un nuovo volume da un volume esistente comporterà la creazione di un nuovo snapshot.

ATTENZIONE	Evitare di clonare una PVC associata a una StorageClass diversa. Eseguire le operazioni di clonazione all'interno della stessa StorageClass per garantire la compatibilità e prevenire comportamenti imprevisti.

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.

Limitare le dimensioni massime dei FlexVol creati da Trident

Per configurare le dimensioni massime per i FlexVol utilizzati come pool per i driver ONTAP-san-Economy e ONTAP-nas-Economy, utilizzare il `limitVolumePoolSize` parametro nella `backend.json` definizione.

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.

IMPORTANTE	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 "Riferimento comando ONTAP".

Limitare l'accesso alle risorse di storage ai membri del cluster Kubernetes

Limitare l'accesso ai volumi NFS, alle LUN iSCSI e alle LUN FC creati da Trident è un componente critico della postura 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

Questa `showmount` funzionalità consente a un client NFS di richiedere all'SVM un elenco di esportazioni NFS disponibili. Un pod implementato nel cluster Kubernetes può emettere un `showmount -e` comando su e ricevere un elenco di mount disponibili, compresi 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

NOTA 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.

La dimensione dei blocchi e la larghezza di banda influiscono direttamente sul numero di IOPS. All'aumentare delle dimensioni dei blocchi, il sistema aumenta la larghezza di banda fino al livello necessario per elaborare blocchi di dimensioni maggiori. All'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 prestazioni.

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.

NOTA 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"
- "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.

Integra Trident con Kubernetes

Per integrare Trident, i seguenti elementi di design e architettura richiedono l'integrazione: Selezione e implementazione dei driver, design della classe di storage, design dei pool virtuali, impatto della rivendicazione del volume persistente (PVC) sul provisioning dello storage, sulle operazioni dei volumi e sull'implementazione dei servizi OpenShift con 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 tramite 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`	Nota a piè di pagina:3[]	Nota a piè di pagina:3[]	Yes[5]	Sì	Nota a piè di pagina:3[]	Sì	Nota a piè di pagina:3[]
`ontap-nas-flexgroup`	Yes[1]	NO	Yes[5]	Sì	Yes[1]	Sì	Yes[1]

Trident offre driver SAN 2 per ONTAP, le cui funzionalità sono mostrate 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	Nota a piè di pagina:3[]	Sì	Nota a piè di pagina:3[]

> Nota a piè di pagina per le tabelle di cui sopra: Nota a piè di pagina:1[]: Non gestito da Trident nota a piè di pagina:2[]: Gestito da Trident, ma non granulare PV nota a piè di pagina:3[]: Non gestito da Trident e non granulare PV nota a piè di pagina:4[]: Supportato per volumi a blocchi grezzi Nota a piè di pagina:5[]: Supportato da 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.

NOTA 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 di pagina: Yes[1]: Non gestito da Trident Yes[2]: Supportato per i volumi di blocchi grezzi

Driver backend Azure NetApp Files

Trident utilizza il `azure-netapp-files` driver per gestire il servizio "Azure NetApp Files".

Ulteriori informazioni su questo driver e su come configurarlo sono disponibili in "Configurazione back-end Trident per Azure NetApp Files".

Driver Azure NetApp Files	Snapshot	Cloni	Multi-attach	QoS	Espandere	Replica
`azure-netapp-files`	Sì	Sì	Sì	Sì	Sì	Yes[1]

> Nota a piè di pagina: Yes[1]: Non gestito da Trident

Driver di backend per Google Cloud NetApp Volumes

Trident supporta il `google-cloud-netapp-volumes` driver per i carichi di lavoro NAS e `google-cloud-netapp-volumes-san` per i carichi di lavoro SAN (iSCSI). Sui cluster OpenShift in esecuzione su GCP, entrambi i driver supportano Google Cloud Workload Identity Federation a partire da Trident 26.10. Ciò include OpenShift Container Platform (OCP, incluse le implementazioni IPI) autogestito e OpenShift Dedicated (OSD).

Per la configurazione del backend e i prerequisiti WIF, vedi "Configurare Google Cloud NetApp Volumes" e "Configura Google Cloud NetApp Volumes per carichi di lavoro SAN".

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 spazio di archiviazione all'insieme di pool che corrispondono a qualsiasi attributo specificato. Il `additionalStoragePools` parametro viene utilizzato per estendere l'insieme di pool utilizzati da Trident per il provisioning insieme all'insieme di pool selezionati dagli attributi e dai `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

I pool virtuali sono disponibili per tutti i backend Trident. È possibile definire pool virtuali per qualsiasi backend, utilizzando qualsiasi driver fornito da 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 archiviazione è configurata con un selettore con le etichette specifiche, Trident sceglie un backend che corrisponde a tutte le etichette del selettore per posizionare il volume. Se le etichette del selettore della classe di archiviazione corrispondono a più pool di archiviazione, Trident sceglierà uno di essi da cui eseguire il provisioning del volume.

Progettazione di un pool virtuale

Durante la creazione di un backend, è generalmente possibile specificare un set di parametri. Era impossibile per l'amministratore 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. Un pool virtuale è un'astrazione di livello introdotta tra il backend e la classe di storage di Kubernetes, in modo che l'amministratore possa definire parametri insieme a etichette a cui è possibile fare riferimento tramite le classi di storage di Kubernetes come selettore, in modo indipendente dal backend. I pool virtuali possono essere definiti per tutti i backend NetApp supportati con Trident. L'elenco include SolidFire/ NetApp HCI, ONTAP e Azure NetApp Files.

NOTA	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 archiviazione richiesta possono influire sul processo decisionale di provisioning 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.

Trident tenterà di corrispondere al protocollo di storage utilizzato con il metodo di accesso specificato secondo la 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.

NOTA	I provisioner in-tree Kubernetes non supportano in questo momento le operazioni di ridimensionamento del volume per NFS, iSCSI o FC PVS. Trident supporta l'espansione di volumi NFS, iSCSI e FC.

I dettagli di connessione del PV non possono essere modificati dopo la creazione.

Creazione di snapshot di volumi on-demand

Trident supporta la creazione di snapshot del volume on-demand e la creazione di PVC dalle 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

Trident supporta anche la creazione di PersistentVolumes dalle istantanee di volume. A tale scopo, è sufficiente creare un'istruzione PersistentVolumeClaim e indicare `datasource` come lo snapshot richiesto da cui creare il volume. Trident gestirà questo PVC creando un volume con i dati presenti sullo 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 Trident o sul cluster Kubernetes, a condizione che l'aggregato di destinazione sia uno a cui ha accesso la SVM utilizzata da Trident. Inoltre, se l'aggregato è stato appena aggiunto alla SVM, sarà necessario aggiornare il backend aggiungendolo nuovamente a Trident. In questo modo, Trident eseguirà il re-inventario della SVM in modo che venga riconosciuto il nuovo aggregato.

Tuttavia, lo spostamento dei volumi tra i backend non è supportato automaticamente da Trident. Si tratta di attività comprese fra SVM dello stesso cluster, fra cluster o in una diversa piattaforma storage (anche se il sistema storage è connesso a Trident).

Se un volume viene copiato in un'altra posizione, è possibile utilizzare la funzione di importazione del volume per importare i volumi correnti in Trident.

Espandere i volumi

Trident supporta il ridimensionamento di PV NFS, iSCSI e FC. Ciò consente agli utenti di ridimensionare i propri volumi direttamente tramite il livello Kubernetes. L'espansione del volume è possibile per tutte le principali piattaforme di storage NetApp , inclusi ONTAP e i backend SolidFire/ NetApp HCI . Per consentire una possibile espansione successiva, impostare `allowVolumeExpansion` A `true` nella StorageClass associata al volume. Ogni volta che è necessario ridimensionare il volume persistente, modificare il `spec.resources.requests.storage` annotazione nella Persistent Volume Claim alla dimensione del volume richiesta. Trident si occuperà automaticamente di ridimensionare il volume sul cluster di archiviazione.

Importare un volume esistente in Kubernetes

L'importazione di volumi consente di importare un volume di archiviazione esistente in un ambiente Kubernetes. Questo è attualmente supportato da `ontap-nas` , `ontap-nas-flexgroup` , `solidfire-san` , E `azure-netapp-files` conducenti. Questa funzionalità è utile quando si trasferisce un'applicazione esistente in Kubernetes o durante scenari di disaster recovery.

Quando si utilizzano ONTAP e driver, utilizzare il comando `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` per importare un volume esistente in Kubernetes e `solidfire-san` gestirlo da Trident. Il file PVC YAML o JSON utilizzato nel comando volume di importazione punta a una classe di archiviazione che identifica Trident come provisioner. 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` viene utilizzato il driver, utilizzare il comando `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` per importare il volume in Kubernetes affinché venga gestito da Trident. Ciò garantisce un riferimento volumetrico univoco.

Quando viene eseguito il comando sopra indicato, Trident trova il volume del backend e ne legge le dimensioni. Aggiungerà automaticamente (e sovrascriverà se necessario) le dimensioni del volume del PVC configurato. 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.

Servizio di registro

La distribuzione e la gestione dello storage per il registro sono state documentate su "netapp.io" in "blog".

Servizio di registrazione

Come gli altri servizi OpenShift, il servizio di logging viene implementato utilizzando Ansible con parametri di configurazione forniti dal file di inventario, ovvero host, forniti al playbook. Sono previsti due metodi di installazione: Distribuzione del logging durante l'installazione iniziale di OpenShift e distribuzione del logging dopo l'installazione di OpenShift.

AVVERTENZA	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.

NOTA	Le tabelle seguenti contengono solo le variabili rilevanti per la configurazione dello storage in relazione al servizio di logging. Puoi trovare altre opzioni in "Documentazione di registrazione di Red Hat OpenShift" che dovrebbero essere riviste, configurate e utilizzate in base alla tua distribuzione.

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.

Variabile	Dettagli
`openshift_logging_storage_host`	Il nome host o l'indirizzo IP dell'host NFS. Tale impostazione deve essere impostata su dataLIF 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 il deploy dopo l'installazione iniziale, il playbook del componente dovrà essere usato da Ansible. Questa procedura può cambiare leggermente con versioni diverse di OpenShift, quindi assicurati di leggere e seguire "Documentazione di Red Hat OpenShift Container Platform 3,11" per la tua versione.

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.

NOTA 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. Questo valore deve essere impostato su dataLIF per la tua 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 crea tutti i PVC necessari e, dopo che Trident ha eseguito il provisioning dello storage per loro, implementa il servizio.

Le variabili di cui sopra e il processo di distribuzione possono cambiare con ogni versione di OpenShift. Assicuratevi di rivedere e seguire "Guida all'implementazione di OpenShift di Red Hat" per la vostra versione affinché sia configurato per il vostro ambiente.

Protezione dei dati e disaster recovery di Trident

Scopri le opzioni di protezione e recovery per Trident e volumi creati con 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 Trident

È possibile creare un backup per ripristinare Trident in caso di emergenza.

Replica Trident

Trident utilizza i CRD Kubernetes per memorizzare e gestire il proprio stato, mentre il cluster etcd Kubernetes memorizza 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 volume

NOTA	NetApp consiglia di proteggere la SVM sul quale si trova FlexVol con una relazione di SnapMirror in un'altra SVM.

Ripristino Trident

Grazie ai Kubernetes CRD e allo snapshot etcd del cluster Kubernetes, puoi ripristinare 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

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 sapere quali opzioni sono state selezionate durante la preformatura Ripristino di SVM mediante 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 di SVM mediante Trident

Trident non rileva automaticamente i guasti della 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.

NOTA	Se specificato `-identity-preserve true`, tutti i volumi con provisioning da Trident iniziano a fornire i dati quando viene attivata la SVM di destinazione.

Replica e recovery dei volumi

Trident non può 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 Trident.

È quindi possibile importare i volumi recuperati in Trident utilizzando "importazione di volumi tridentctl".

NOTA	L'importazione di volumi è supportata con il driver `ontap-san-economy`. L'importazione di Qtree con il driver `ontap-nas-economy` è una funzionalità Tech Preview e non è supportata per l'uso in produzione. L'importazione non è supportata con il driver `ontap-flexgroup-economy`. Per i dettagli, vedi "Importa i volumi Trident".

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"

Automazione del failover delle applicazioni stateful con Trident

La funzionalità di distacco forzato di Trident consente di staccare automaticamente i volumi dai nodi non integri in un cluster Kubernetes, prevenendo il danneggiamento dei dati e garantendo la disponibilità delle applicazioni. Questa funzionalità è particolarmente utile negli scenari in cui i nodi non rispondono più o vengono disconnessi per manutenzione.

Dettagli sulla forza di distacco

Il distacco forzato è disponibile per `ontap-san` , `ontap-san-economy` , `ontap-nas` , E `ontap-nas-economy` soltanto. Prima di abilitare la disconnessione forzata, è necessario abilitare l'arresto non regolare del nodo (NGNS) sul cluster Kubernetes. NGNS è abilitato per impostazione predefinita per Kubernetes 1.28 e versioni successive. Per ulteriori informazioni, fare riferimento a "Kubernetes: Shutdown del nodo non aggraziato" .

NOTA	Quando si utilizza il `ontap-nas` driver OR `ontap-nas-economy`, è necessario impostare il `autoExportPolicy` parametro nella configurazione backend in `true` modo che Trident possa limitare l'accesso dal nodo Kubernetes con il tag applicato utilizzando policy di esportazione gestite.

ATTENZIONE	Poiché Trident fa affidamento su Kubernetes NGNS, non rimuovere i `out-of-service` tag da un nodo non integro fino a quando tutti i carichi di lavoro non tollerabili non vengono ripianificati. L'applicazione o la rimozione sconsiderata della contaminazione può compromettere la protezione dei dati back-end.

Quando l'amministratore del cluster Kubernetes ha applicato il `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` tag al nodo ed `enableForceDetach` è impostato su `true`, Trident determinerà lo stato del nodo e:

1. Interrompere l'accesso I/O backend per i volumi montati su quel nodo.
2. Contrassegnare l'oggetto nodo Trident come `dirty` (non sicuro per le nuove pubblicazioni).

NOTA	Il controller Trident rifiuterà le nuove richieste di volume di pubblicazione finché il nodo non viene riqualificato (dopo essere stato contrassegnato come `dirty`) dal pod di nodo Trident. Tutti i carichi di lavoro pianificati con un PVC montato (anche dopo che il nodo del cluster è integro e pronto) non saranno accettati fino a quando Trident non sarà in grado di verificare il nodo `clean` (sicuro per le nuove pubblicazioni).

Quando l'integrità del nodo viene ripristinata e il tag viene rimosso, Trident:

1. Identificare e pulire i percorsi pubblicati obsoleti sul nodo.
2. Se il nodo si trova in uno `cleanable` stato (il tag out-of-service è stato rimosso e il nodo è nello `Ready` stato) e tutti i percorsi obsoleti e pubblicati sono puliti, Trident riammetterà il nodo come `clean` e consentirà ai nuovi volumi pubblicati di accedere al nodo.

Dettagli sul failover automatico

È possibile automatizzare il processo di distacco forzato tramite l'integrazione con"operatore di controllo dello stato di salute del nodo (NHC)" . Quando si verifica un errore di nodo, NHC attiva la riparazione del nodo Trident (TNR) e forza automaticamente il distacco creando un CR TridentNodeRemediation nello spazio dei nomi di Trident che definisce il nodo in errore. Il TNR viene creato solo in caso di guasto del nodo e rimosso da NHC una volta che il nodo torna online o viene eliminato.

Processo di rimozione del pod del nodo non riuscito

Il failover automatico seleziona i carichi di lavoro da rimuovere dal nodo in errore. Quando viene creato un TNR, il controller TNR contrassegna il nodo come sporco, impedendo la pubblicazione di nuovi volumi e inizia a rimuovere i pod supportati dalla funzione di distacco forzato e i relativi allegati di volume.

Tutti i volumi/PVC supportati da force-detach sono supportati da automatizzate-failover:

- Volumi NAS e NAS-economy che utilizzano criteri di esportazione automatica (SMB non è ancora supportato).
- Volumi SAN e SAN-economy.

Fare riferimento aDettagli sulla forza di distacco .

Comportamento predefinito:

- I pod che utilizzano volumi supportati da force-detach vengono rimossi dal nodo in errore. Kubernetes li riprogrammerà su un nodo sano.
- I pod che utilizzano un volume non supportato dal distacco forzato, inclusi i volumi non Trident, non vengono rimossi dal nodo in errore.
- I pod senza stato (non PVC) non vengono rimossi dal nodo non riuscito, a meno che l'annotazione del pod `trident.netapp.io/podRemediationPolicy: delete` è impostato.

Sostituire il comportamento di rimozione del pod:

Il comportamento di rimozione del pod può essere personalizzato utilizzando un'annotazione del pod: `trident.netapp.io/podRemediationPolicy[retain, delete]` . Queste annotazioni vengono

esaminate e utilizzate quando si verifica un failover. Applicare annotazioni alla specifica del pod di distribuzione/replicaset di Kubernetes per evitare che l'annotazione scompaia dopo un failover:

- `retain`- Il pod NON verrà rimosso dal nodo in errore durante un failover automatico.
- `delete`- Il pod verrà rimosso dal nodo in errore durante un failover automatico.

Queste annotazioni possono essere applicate a qualsiasi pod.

ATTENZIONE	• Le operazioni di I/O verranno bloccate solo sui nodi non riusciti per i volumi che supportano il distacco forzato. • Per i volumi che non supportano il distacco forzato, esiste il rischio di danneggiamento dei dati e di problemi di collegamento multiplo.

TridentNodeRemediation CR

Il CR TridentNodeRemediation (TNR) definisce un nodo non riuscito. Il nome del TNR è il nome del nodo non riuscito.

Esempio TNR:

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

Stati TNR: utilizzare i seguenti comandi per visualizzare lo stato dei TNR:
`kubectl get tnr <name> -n <trident-namespace>`

I TNR possono trovarsi in uno dei seguenti stati:

- *Rimediando*:
  - Interrompere l'accesso I/O backend per i volumi supportati da force-detach montati su quel nodo.
  - L'oggetto nodo Trident è contrassegnato come sporco (non sicuro per nuove pubblicazioni).
  - Rimuovi i pod e gli allegati di volume dal nodo
- *NodeRecoveryPending*:
  - Il controller attende che il nodo torni online.
  - Una volta che il nodo è online, publish-enforcement garantirà che il nodo sia pulito e pronto per le nuove pubblicazioni di volumi.
- Se il nodo viene eliminato da K8s, il controller TNR rimuoverà il TNR e cesserà la riconciliazione.
- *Riuscito*:
  - Tutti i passaggi di ripristino e ripristino dei nodi sono stati completati con successo. Il nodo è pulito e pronto per la pubblicazione di nuovi volumi.
- *Non riuscito*:
  - Errore irrecuperabile. I motivi dell'errore vengono impostati nel campo status.message del CR.

Abilitazione del failover automatico

Prerequisiti:

- Prima di abilitare il failover automatico, assicurarsi che la disconnessione forzata sia abilitata. Per maggiori informazioni, fare riferimento aDettagli sulla forza di distacco .
- Installare il controllo dello stato del nodo (NHC) nel cluster Kubernetes.
  - "Installa operator-sdk".
  - Installare Operator Lifecycle Manager (OLM) nel cluster se non è già installato: `operator-sdk olm install` .
  - Installa l'operatore di controllo dello stato del nodo: `kubectl create -f https://operatorhub.io/install/node-healthcheck-operator.yaml` .

NOTA	È anche possibile utilizzare metodi alternativi per rilevare gli errori dei nodi, come specificato nella sezione Integrazione di soluzioni personalizzate per il controllo dello stato dei nodi qui sotto.

Vedere"Operatore di controllo dello stato del nodo" per maggiori informazioni.

Fasi

1. Creare un CR NodeHealthCheck (NHC) nello spazio dei nomi Trident per monitorare i nodi worker nel cluster. Esempio:

```
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: <CR name>
spec:
  selector:
    matchExpressions:
      - key: node-role.kubernetes.io/control-plane
        operator: DoesNotExist
      - key: node-role.kubernetes.io/master
        operator: DoesNotExist
  remediationTemplate:
    apiVersion: trident.netapp.io/v1
    kind: TridentNodeRemediationTemplate
    namespace: <Trident installation namespace>
    name: trident-node-remediation-template
  minHealthy: 0 # Trigger force-detach upon one or more node failures
  unhealthyConditions:
    - type: Ready
      status: "False"
      duration: 0s
    - type: Ready
      status: Unknown
```

```
      duration: 0s
```

2. Applicare il controllo di integrità del nodo CR nel `trident` spazio dei nomi.

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

Il CR sopra indicato è configurato per monitorare i nodi worker K8 per le condizioni del nodo Pronto: falso e Sconosciuto. Il failover automatico verrà attivato quando un nodo passa allo stato Pronto: falso o Pronto: sconosciuto.

IL `unhealthyConditions` nel CR utilizza un periodo di grazia di 0 secondi. Ciò fa sì che il failover automatico venga attivato immediatamente quando K8s imposta la condizione del nodo Ready: false, che viene impostata dopo che K8s perde l'heartbeat da un nodo. K8s ha un'attesa predefinita di 40 secondi dopo l'ultimo battito cardiaco prima di impostare Ready: false. Questo periodo di grazia può essere personalizzato nelle opzioni di distribuzione di K8.

Per ulteriori opzioni di configurazione, fare riferimento a"Documentazione di Node-Healthcheck-Operator" .

Informazioni aggiuntive sulla configurazione

Quando Trident viene installato con la funzione force-detach abilitata, vengono create automaticamente due risorse aggiuntive nello spazio dei nomi Trident per facilitare l'integrazione con NHC: TridentNodeRemediationTemplate (TNRT) e ClusterRole.

TridentNodeRemediationTemplate (TNRT):

Il TNRT funge da modello per il controller NHC, che utilizza il TNRT per generare risorse TNR secondo necessità.

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediationTemplate
metadata:
  name: trident-node-remediation-template
  namespace: trident
spec:
  template:
    spec: {}
```

RuoloCluster:

Durante l'installazione, quando è abilitato il distacco forzato, viene aggiunto anche un ruolo cluster. Ciò fornisce autorizzazioni NHC ai TNR nello spazio dei nomi Trident .

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    rbac.ext-remediation/aggregate-to-ext-remediation: "true"
  name: tridentnoderemediation-access
```

```
rules:
- apiGroups:
  - trident.netapp.io
  resources:
  - tridentnoderemediationtemplates
  - tridentnoderemediations
  verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
```

Aggiornamenti e manutenzione del cluster K8s

Per evitare eventuali failover, sospendere il failover automatico durante la manutenzione o gli aggiornamenti di K8, quando è previsto che i nodi si fermino o si riavviino. È possibile mettere in pausa il CR NHC (descritto sopra) applicando una patch al suo CR:

```
kubectl patch NodeHealthCheck <cr-name> --patch
'{"spec":{"pauseRequests":["<description-for-reason-of-pause>"]}}' --type=merge
```

In questo modo si sospende il failover automatico. Per riattivare il failover automatico, rimuovere pauseRequests dalla specifica una volta completata la manutenzione.

Limitazioni

- Le operazioni di I/O vengono impedite solo sui nodi non riusciti per i volumi supportati da force-detach. Vengono rimossi automaticamente solo i pod che utilizzano volumi/PVC supportati da force-detach.
- Il failover automatico e il distacco forzato vengono eseguiti all'interno del pod del controller Trident. Se il nodo che ospita il controller Trident si guasta, il failover automatico verrà ritardato finché K8s non sposterà il pod su un nodo funzionante.

Integrazione di soluzioni personalizzate per il controllo dello stato dei nodi

È possibile sostituire Node Healthcheck Operator con strumenti alternativi di rilevamento degli errori dei nodi per attivare il failover automatico. Per garantire la compatibilità con il meccanismo di failover automatico, la soluzione personalizzata deve:

- Crea un TNR quando viene rilevato un errore del nodo, utilizzando il nome del nodo in errore come nome CR del TNR.
- Eliminare il TNR quando il nodo è stato ripristinato e il TNR è nello stato Riuscito.

Sicurezza

Raccomandazioni di sicurezza di Trident

Utilizzare i consigli elencati di seguito per assicurarsi che l'installazione di Trident sia sicura.

Eseguire Trident nel proprio namespace

È importante impedire ad applicazioni, amministratori dell'applicazione, utenti e applicazioni di gestione di accedere alle definizioni di oggetti Trident o ai pod, per garantire uno storage affidabile e bloccare le potenziali attività pericolose.

Per separare le altre applicazioni e gli utenti da Trident, installare sempre Trident nel proprio spazio dei nomi Kubernetes (`trident`). Inserendo Trident nel proprio namespace, solo il personale amministrativo di Kubernetes potrà accedere al pod Trident e agli artefatti (come ad esempio backend e CHAP secrets, se applicabili) memorizzati negli oggetti CRD con nome. È necessario assicurarsi che solo gli amministratori possano accedere allo spazio dei nomi Trident e quindi all' `tridentctl`applicazione.

Utilizza l'autenticazione CHAP con i backend SAN ONTAP

Trident supporta l'autenticazione basata su CHAP per i carichi di lavoro SAN ONTAP (mediante `ontap-san` e `ontap-san-economy` driver). NetApp consiglia di utilizzare il protocollo CHAP bidirezionale con Trident per l'autenticazione tra un host e il backend dello storage.

Per i backend ONTAP che utilizzano i driver di archiviazione SAN, Trident può impostare il CHAP bidirezionale e gestire i nomi utente e i segreti CHAP tramite `tridentctl`. Fare riferimento a "Prepararsi a configurare il backend con i driver SAN ONTAP" per informazioni sulla configurazione del protocollo CHAP in Trident 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. Trident utilizza un oggetto segreto che include due password CHAP per tenant. Quando Trident viene installato, gestisce i segreti CHAP e li memorizza in un `tridentvolume` oggetto CR per il PV corrispondente. Quando si crea un PV, Trident utilizza i segreti CHAP per avviare una sessione iSCSI e comunicare con il sistema NetApp HCI e SolidFire tramite CHAP.

NOTA I volumi creati da Trident non sono associati ad alcun gruppo di accesso ai volumi.

USA 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 abilitato sul backend, qualsiasi volume sottoposto a provisioning in Trident sarà abilitato NAE.
  - Puoi impostare il flag di crittografia NVE su `""` per creare volumi abilitati per NAE.
- Se NAE non è abilitato sul back-end, qualsiasi volume con provisioning in Trident sarà abilitato NVE, a meno che il flag di crittografia NVE non sia impostato su `false` (il valore predefinito) nella configurazione di back-end.

NOTA I volumi creati in Trident su un back-end abilitato per NAE devono essere crittografati 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.
- L'impostazione del flag di crittografia NVE su `false` un backend abilitato per NAE crea un volume abilitato per NAE. Non è possibile disattivare la crittografia NAE impostando il flag di crittografia NVE su `false`.

- Puoi creare manualmente un volume NVE in 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"

Trident Linux Unified Key Setup (LUKS)

Linux Unified Key Setup (LUKS) è una specifica per la crittografia del disco. Trident utilizza LUKS per fornire la crittografia lato host per volume per i volumi ONTAP SAN e ONTAP SAN ECONOMY, con supporto per la rotazione della passphrase e l'espansione del volume.

In Trident, i volumi crittografati con LUKS utilizzano il Cypher e la modalità aes-xts-plain64, come consigliato da "NIST".

NOTA La crittografia LUKS non è supportata per i sistemi ASA r2. Per informazioni sui sistemi ASA r2, vedere "Informazioni sui sistemi di storage ASA R2" .

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 prestazioni, NetApp consiglia ai nodi di lavoro di supportare le nuove istruzioni AES-NI (Advanced Encryption Standard New Instructions). 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
```

Usa LUKS con volumi raw a blocchi RWX

A partire da Trident 26.10, puoi utilizzare la crittografia LUKS con volumi a blocchi raw basati su iSCSI collegati a più nodi Kubernetes tramite la modalità di accesso `ReadWriteMany` (RWX). Questo è utile per carichi di lavoro come le macchine virtuali KubeVirt che devono collegare lo stesso volume crittografato a un nodo di destinazione mentre è ancora collegato al nodo di origine durante la migrazione live.

Il supporto RWX LUKS è limitato ai volumi a blocchi raw. I volumi RWX LUKS in modalità filesystem non sono supportati. L'applicazione è responsabile del coordinamento dell'accesso simultaneo ai dati decrittografati; Trident coordina solo le operazioni LUKS.

Configura LUKS sul backend, StorageClass e il Secret della passphrase come descritto in Attivare la crittografia LUKS. Quindi crea un PVC con `volumeMode: Block` e `ReadWriteMany`:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-rwx-pvc
  namespace: my-app
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  storageClassName: san-luks-rwx
  resources:
    requests:
      storage: 10Gi
```

I pod su più nodi possono collegare questo PVC come dispositivo a blocchi raw. Trident formatta il dispositivo con LUKS2 al primo collegamento e apre il volume crittografato esistente sui nodi successivi. Le operazioni LUKS esistenti, tra cui la rotazione della passphrase e l'espansione del volume, sono supportate.

Perché Trident serializza le operazioni LUKS

LUKS2 memorizza copie ridondanti della sua intestazione di crittografia all'inizio del dispositivo a blocchi. `cryptsetup`Può riparare un'intestazione quando rileva che le copie differiscono, anche durante comandi che sembrano di sola lettura. Il suo meccanismo di blocco è locale a un nodo e non può coordinare operazioni su altri nodi. Se un nodo sta aggiornando l'intestazione mentre un altro esegue `cryptsetup`, il secondo nodo potrebbe attivare la riparazione dell'intestazione e interrompere l'aggiornamento, corrompendo potenzialmente l'intestazione.

Per impedire l'accesso simultaneo all'intestazione, Trident utilizza un Lease di Kubernetes per ogni volume RWX LUKS. Prima che un nodo esegua qualsiasi `cryptsetup` comando per il volume, deve detenere il Lease. Trident rilascia il Lease dopo un'operazione riuscita. Se un altro nodo richiede un'operazione mentre il Lease è detenuto, Trident restituisce un errore recuperabile e Kubernetes riprova in seguito. Le operazioni su

volumi diversi sono indipendenti.

Se un' `cryptsetup`operazione fallisce, Trident mantiene intenzionalmente il Lease in modo che solo lo stesso nodo possa riprovare. Un evento di avviso con motivo `LeaseRetained`registra l'errore. I Lease non scadono automaticamente; un amministratore dovrebbe rimuovere un Lease solo dopo aver verificato che nessuna `cryptsetup` operazione sia in esecuzione sul nodo che lo detiene. Per le procedure di ripristino, vedi "Risolvere un Lease bloccato per i volumi RWX Linux Unified Key Setup (LUKS)".

Poiché i nodi si alternano nell'apertura del volume, l'avvio dei carichi di lavoro su molti nodi diversi può richiedere più tempo rispetto a un volume non crittografato. Kubernetes riprova gli errori di blocco temporanei al termine dell'operazione corrente. I pod possono rimanere in `ContainerCreating` più a lungo durante le scalabilità orizzontali su larga scala; i carichi di lavoro sullo stesso nodo possono condividere un'unica mappatura del dispositivo aperto.

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` su(`true` sul backend. L' `luksEncryption`opzione indica a Trident se il volume è (`true`compatibile con LUKS o non compatibile con 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'
```

Configurazione PVC per l'importazione di volumi LUKS

Per importare volumi LUKS in modo dinamico, impostare l'annotazione `trident.netapp.io/luksEncryption` su `true` e includere una classe di storage abilitata LUKS nel PVC, come illustrato in questo esempio.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: luks-pvc
  namespace: trident
```

```
  annotations:
    trident.netapp.io/luksEncryption: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: luks-sc
```

Ruotare una passphrase LUKS

È possibile ruotare la passphrase LUKS e confermare la rotazione.

ATTENZIONE	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, Trident confronta la passphrase LUKS del 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.

NOTA	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.

Risolvere un Lease bloccato per i volumi RWX Linux Unified Key Setup (LUKS)

Un volume ReadWriteMany (RWX) Linux Unified Key Setup (LUKS) può bloccarsi se un nodo si arresta mentre detiene il relativo Lease di Kubernetes. Conferma che non `cryptsetup` sia ancora in corso alcuna operazione prima di eliminare il Lease; eliminarlo durante un'operazione può mettere a rischio l'intestazione LUKS.

Comprendi il lease

Trident utilizza un Lease di Kubernetes per garantire che solo un nodo alla volta esegua `cryptsetup` comandi su un volume LUKS RWX. I Lease vengono creati nello spazio dei nomi Trident e il nome del Lease è

il nome del PersistentVolume (PV). L'identità del titolare identifica il nodo che esegue l'operazione.

Trident elimina il Lease dopo un'operazione riuscita. Se un'operazione fallisce, Trident mantiene il Lease in modo che solo lo stesso nodo possa riprovare. I Lease non scadono né si rinnovano automaticamente. Un evento di avviso con motivo `LeaseRetained` include i dettagli su un'operazione non riuscita.

Identifica un Lease potenzialmente bloccato

I sintomi possono includere pod bloccati in `ContainerCreating` o `Terminating`, mancata espansione del volume o errori di blocco temporanei.

1. Individua il PVC utilizzato da un pod interessato:

```
kubectl get pod -n <app-namespace> <pod-name> \
  -o jsonpath='{.spec.volumes[*].persistentVolumeClaim.claimName}'
```

2. Trova il PV associato al PVC:

```
kubectl get pvc -n <app-namespace> <pvc-name> \
  -o jsonpath='{.spec.volumeName}'
```

3. Elenca i Lease nel namespace Trident e identifica il Lease con il nome del PV:

```
kubectl get lease -n <trident-namespace>
```

4. Esamina il Lease corrispondente, inclusi il titolare, l'età e gli eventi:

```
kubectl describe lease -n <trident-namespace> <pv-name>
```

`cryptsetup` le operazioni si completano normalmente in pochi secondi. Un Lease più vecchio di 5-10 minuti può indicare un deadlock. Controlla gli `LeaseRetained` eventi per individuare errori ripetuti, identifica il nodo detentore e verifica lo stato di salute del nodo:

```
kubectl get nodes
kubectl get pods -A -o wide --field-selector spec.nodeName=<lease-holder-
node>
```

Se il nodo è integro e il Lease non presenta eventi di avviso, esamina i log del pod del nodo Trident sul nodo detentore prima di concludere che il Lease sia bloccato:

```
kubectl get pods -n <trident-namespace> -o wide \
  --field-selector spec.nodeName=<lease-holder-node>
kubectl logs -n <trident-namespace> <trident-node-pod> -c trident-main
```

Se il problema del nodo o della rete sottostante può essere risolto, fallo prima. Il Lease può risolversi quando Trident completa correttamente l'operazione.

Risolvi in modo sicuro il Lease bloccato

IMPORTANTE	Non eliminare un Lease mentre il suo detentore potrebbe essere ancora impegnato in una `cryptsetup`operazione. Comandi `cryptsetup` concorrenti provenienti da nodi diversi e non coordinati dal Lease possono corrompere l'intestazione LUKS.

Usa la soluzione correttiva che corrisponde alla condizione del nodo:

- **Il nodo titolare si è guastato ed è spento:** identifica i Lease detenuti da quel nodo ed elimina il Lease interessato. Se il nodo viene ripristinato prima che i suoi carichi di lavoro vengano riprogrammati, consenti a Trident di completare l'operazione e rilasciare il Lease. Se i carichi di lavoro sono già stati spostati, attendi che i relativi pod sul nodo ripristinato terminino completamente, verifica che il deadlock persista e quindi rimuovi il Lease obsoleto.
- **Il nodo che detiene il controllo ha perso la connettività al control plane:** il nodo potrebbe ancora eseguire carichi di lavoro. Preferisci arrestare il nodo prima di eliminare il Lease. Se il nodo rimane accessibile, arresta il container del nodo Trident su quel nodo o verifica dai suoi log che non possa acquisire o eliminare il Lease e che non sia in esecuzione `cryptsetup`. Trident richiede l'accesso all'API di Kubernetes per acquisire il Lease prima di ogni operazione; dopo che la connettività viene persa, non può avviare un'altra operazione. Attendi almeno cinque minuti dalla creazione del Lease prima di considerarne l'eliminazione, per consentire il completamento di eventuali comandi in corso.
- **Il nodo detentore e la rete sono integri:** Esamina gli eventi di avviso di Lease e i log del nodo di Trident per identificare l'operazione che non riesce. Se devi spostare il workload interessato su un altro nodo, metti il nodo detentore in cordon, elimina i pod dell'applicazione interessati senza forzarne l'eliminazione, attendi che terminino e rimuovi il cordon dal nodo. Elimina il Lease solo dopo aver confermato che l'operazione originale non è più in esecuzione.

Elimina il Lease solo dopo aver verificato che sia sicuro farlo:

```
kubectl delete lease -n <trident-namespace> <pv-name>
```

Dopo la correzione, monitora i pod e le operazioni sui volumi interessati. Se il Lease viene conservato di nuovo, esamina l'errore cryptsetup segnalato negli eventi del Lease e risolvi il problema sottostante.

Crittografia in transito Kerberos di Trident

Utilizzando la crittografia in-flight Kerberos, puoi migliorare la sicurezza dell'accesso ai dati abilitando la crittografia per il traffico tra il cluster gestito e il backend dello storage.

Trident supporta la crittografia Kerberos per ONTAP come backend di storage:

- **On-premise ONTAP** - Trident supporta la crittografia Kerberos su connessioni NFSv3 e NFSv4 da cluster Red Hat OpenShift e 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 in sede

È possibile abilitare la crittografia Kerberos sul traffico di storage tra il cluster gestito e un backend di storage ONTAP on-premise.

NOTA	La crittografia Kerberos per il traffico NFS con backend di archiviazione ONTAP in sede è supportata solo utilizzando il `ontap-nas` driver di archiviazione.

Prima di iniziare

- 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 un dataLIF" per le istruzioni.
- Assicurarsi che i volumi NFSv4 utilizzati con la crittografia Kerberos siano configurati correttamente. Fare riferimento alla sezione NetApp NFSv4 Domain Configuration (pagina 13) di "NetApp Miglioramenti NFSv4 e Guida alle Best Practice".

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

È possibile creare una configurazione backend dello storage Trident che include la 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. Per istruzioni, fare riferimento alla "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 Trident sul cluster gestito di Red Hat OpenShift.
- 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".
- Assicurarsi che i volumi NFSv4 utilizzati con la crittografia Kerberos siano configurati correttamente. Fare riferimento alla sezione NetApp NFSv4 Domain Configuration (pagina 13) di "NetApp Miglioramenti NFSv4 e Guida alle Best Practice".

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. Per istruzioni, fare riferimento alla "Provisioning di un volume".

Proteggi le applicazioni con Trident Protect

Scopri di più su Trident Protect

NetApp Trident Protect offre funzionalità avanzate di gestione dei dati delle applicazioni che migliorano la funzionalità e la disponibilità delle applicazioni Kubernetes con stato supportate dai sistemi di storage NetApp ONTAP e dal provisioner di storage NetApp Trident CSI. Trident Protect semplifica la gestione, la protezione e lo spostamento dei carichi di lavoro containerizzati tra cloud pubblici e ambienti on-premise. Offre inoltre funzionalità di automazione tramite API e CLI.

È possibile proteggere le applicazioni con Trident Protect creando risorse personalizzate (CR) o utilizzando la CLI Trident Protect.

Quali sono le prossime novità?

Puoi informarti sui requisiti Trident Protect prima di installarlo:

- "Requisiti Trident Protect"

Installa Trident Protect

Requisiti Trident Protect

Per iniziare, verifica la prontezza del tuo ambiente operativo, dei cluster applicativi, delle applicazioni e delle licenze. Assicurati che il tuo ambiente soddisfi questi requisiti per distribuire e utilizzare Trident Protect.

Compatibilità del cluster Kubernetes Trident Protect

Trident Protect è compatibile con un'ampia gamma di offerte Kubernetes completamente gestite e autogestite, tra cui:

- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- Servizio Kubernetes di Microsoft Azure (AKS)
- Red Hat OpenShift
- SUSE Harvester 1.7.0 (ONTAP iSCSI)
- SUSE Rancher
- Portfolio VMware Tanzu
- Kubernetes upstream

NOTA

- I backup Trident Protect sono supportati solo sui nodi di elaborazione Linux. I nodi di elaborazione Windows non sono supportati per le operazioni di backup.
- Assicurarsi che il cluster su cui si installa Trident Protect sia configurato con un controller snapshot in esecuzione e i relativi CRD. Per installare un controller snapshot, fare

riferimento a "queste istruzioni" .

- Assicurarsi che esista almeno un VolumeSnapshotClass. Per ulteriori informazioni, consultare "VolumeSnapshotClass".
- Per installare Trident Protect è necessario Helm 4.x o versioni successive.

Compatibilità del backend di archiviazione Trident Protect

Trident Protect supporta i seguenti backend di archiviazione:

- Amazon FSX per NetApp ONTAP
- Cloud Volumes ONTAP
- Array storage ONTAP
- Google Cloud NetApp Volumes
- Azure NetApp Files

Verificare che lo storage backend soddisfi i seguenti requisiti:

- Assicurarsi che lo storage NetApp connesso al cluster utilizzi Trident 24.02 o una versione successiva (si consiglia Trident 24.10).
- Verificare di disporre di un back-end dello storage NetApp ONTAP.
- Verificare di aver configurato un bucket dello storage a oggetti per la memorizzazione dei backup.
- Crea tutti gli spazi dei nomi delle applicazioni che intendi utilizzare per le applicazioni o per le operazioni di gestione dei dati delle applicazioni. Trident Protect non crea questi namespace per te; se specifichi uno namespace inesistente in una risorsa personalizzata, l'operazione non riuscirà.

Per i volumi nas-Economy

Trident Protect supporta le operazioni di backup e ripristino sui volumi nas-economy. Snapshot, cloni e replica SnapMirror su volumi nas-economy non sono attualmente supportati. È necessario abilitare una directory snapshot per ogni volume nas-economy che si intende utilizzare con Trident Protect.

NOTA

Alcune applicazioni non sono compatibili con volumi che utilizzano una directory snapshot. Per queste applicazioni, è necessario nascondere la directory dello snapshot eseguendo il seguente comando nel sistema di archiviazione ONTAP:

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

Puoi abilitare la directory dello snapshot eseguendo il seguente comando per ogni volume di economia nas, sostituendo `<volume-UUID>` con l'UUID del volume che desideri modificare:

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

NOTA

Per impostazione predefinita, è possibile abilitare le directory snapshot per i nuovi volumi impostando l'opzione di configurazione back-end Trident `snapshotDir` su `true`. I volumi

esistenti non vengono influenzati.

Protezione dei dati con le macchine virtuali KubeVirt

Trident Protect fornisce funzionalità di blocco e sblocco del file system per le macchine virtuali KubeVirt durante le operazioni di protezione dei dati per garantire la coerenza dei dati. Il metodo di configurazione e il comportamento predefinito per le operazioni di blocco delle VM variano a seconda delle versioni Trident Protect, con le versioni più recenti che offrono una configurazione semplificata tramite i parametri del grafico Helm.

NOTA Durante le operazioni di ripristino, qualsiasi `VirtualMachineSnapshots` creati per una macchina virtuale (VM) non vengono ripristinati.

▼ **Trident Protect 25.10 e versioni successive**

Trident Protect blocca e sblocca automaticamente i file system KubeVirt durante le operazioni di protezione dei dati per garantire la coerenza. A partire da Trident Protect 25.10, è possibile disattivare questo comportamento utilizzando `vm.freeze` parametro durante l'installazione della carta Helm. Il parametro è abilitato per impostazione predefinita.

```
helm install ... --set vm.freeze=false ...
```

▼ **Trident Protect dal 24.10.1 al 25.06**

A partire da Trident Protect 24.10.1, Trident Protect blocca e sblocca automaticamente i file system KubeVirt durante le operazioni di protezione dei dati. Facoltativamente, è possibile disattivare questo comportamento automatico utilizzando il seguente comando:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=false -n trident-protect
```

▼ **Trident Protect 24.10**

Trident Protect 24.10 non garantisce automaticamente uno stato coerente per i file system delle VM KubeVirt durante le operazioni di protezione dei dati. Se si desidera proteggere i dati della VM KubeVirt utilizzando Trident Protect 24.10, è necessario abilitare manualmente la funzionalità di congelamento/scongelamento per i file system prima dell'operazione di protezione dei dati. Ciò garantisce che i file system siano in uno stato coerente.

È possibile configurare Trident Protect 24.10 per gestire il blocco e lo sblocco del file system della VM durante le operazioni di protezione dei dati "configurazione della virtualizzazione" e quindi utilizzare il seguente comando:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

Per sapere come definire le applicazioni e ripristinare le VM con Trident Protect, vedi "Proteggi le macchine virtuali KubeVirt con Trident Protect".

Requisiti per la replica SnapMirror

La replica NetApp SnapMirror è disponibile per l'uso con Trident Protect per le seguenti soluzioni ONTAP :

- Sistemi NetApp FAS, AFF e ASA on-premises. La replica SnapMirror con Trident protect non è attualmente supportata per i sistemi ASA r2.
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSX per NetApp ONTAP

Requisiti del cluster di ONTAP per la replica SnapMirror

Assicurati che il tuo cluster ONTAP soddisfi i seguenti requisiti se intendi utilizzare la replica SnapMirror:

- * NetApp Trident*: NetApp Trident deve essere presente sia sul cluster Kubernetes di origine che su quello di destinazione che utilizzano ONTAP come backend. Trident Protect supporta la replica con la tecnologia NetApp SnapMirror utilizzando classi di archiviazione supportate dai seguenti driver:
  - `ontap-nas` : NFS
  - `ontap-san` : iSCSI
  - `ontap-san` : FC
  - `ontap-san` : NVMe/TCP (richiede almeno la versione ONTAP 9.15.1)
- **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.

  A partire da ONTAP 9.10.1, tutte le licenze vengono fornite come file di licenza NetApp (NLF), che è un singolo file che abilita più funzioni. Per ulteriori informazioni, fare riferimento "Licenze incluse con ONTAP ONE" a.

  NOTA È supportata solo la protezione asincrona SnapMirror.

Considerazioni sul peering per la replica SnapMirror

Assicurati che il tuo ambiente soddisfi i seguenti requisiti se intendi utilizzare il peering di back-end dello storage:

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

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

- **NetApp Trident e SVM**: le SVM remote peered devono essere disponibili per NetApp Trident sul cluster di destinazione.
- **Backend gestiti**: è necessario aggiungere e gestire i backend di archiviazione ONTAP in Trident Protect per creare una relazione di replica.

Configurazione Trident / ONTAP per la replica SnapMirror

Trident Protect richiede la configurazione di almeno un backend di archiviazione che supporti la replica sia per i cluster di origine che per quelli di destinazione. Se i cluster di origine e di destinazione sono gli stessi, l'applicazione di destinazione dovrebbe utilizzare un backend di archiviazione diverso da quello dell'applicazione di origine per ottenere la migliore resilienza.

Requisiti del cluster Kubernetes per la replica SnapMirror

Assicurati che i tuoi cluster Kubernetes soddisfino i seguenti requisiti:

- **Accessibilità ad AppVault**: sia i cluster di origine che quelli di destinazione devono avere accesso alla rete per leggere e scrivere su AppVault per la replica degli oggetti applicativi.
- **Connettività di rete**: configura le regole del firewall, le autorizzazioni dei bucket e le liste consentite di IP per abilitare la comunicazione tra entrambi i cluster e AppVault attraverso le WAN.

NOTA	Molti ambienti aziendali implementano rigide policy firewall sulle connessioni WAN. Verificare questi requisiti di rete con il team dell'infrastruttura prima di configurare la replica.

Installa e configura Trident Protect

Se il tuo ambiente soddisfa i requisiti per Trident Protect, puoi seguire questi passaggi per installare Trident Protect sul tuo cluster. Puoi ottenere Trident Protect da NetApp oppure installarlo dal tuo registro privato. L'installazione da un registro privato è utile se il cluster non riesce ad accedere a Internet.

Installa Trident Protect

Installa Trident Protect da NetApp

Fasi

1. Aggiungere il repository Trident Helm:

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

2. Utilizzare Helm per installare Trident Protect. Sostituire `<name-of-cluster>` con un nome cluster, che verrà assegnato al cluster e utilizzato per identificare i backup e gli snapshot del cluster:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2610.0 --create
-namespace --namespace trident-protect
```

3. Facoltativamente, per abilitare la registrazione del debug (consigliata per la risoluzione dei problemi), utilizzare:

```
helm install trident-protect netapp-trident-protect/trident-protect
```

```
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2610.0 --create-namespace --namespace trident-protect
```

La registrazione del debug aiuta NetApp a risolvere i problemi senza dover modificare il livello di registrazione o riprodurre i problemi.

Installa Trident Protect da un registro privato

È possibile installare Trident Protect da un registro di immagini privato se il cluster Kubernetes non è in grado di accedere a Internet. In questi esempi, sostituisci i valori tra parentesi con le informazioni provenienti dal tuo ambiente:

Fasi

1. Estrarre le seguenti immagini sul computer locale, aggiornare i tag e quindi inviarle al registro privato:

```
docker.io/netapp/controller:26.10.1
docker.io/netapp/trident-protect-utils:v6.0.0
docker.io/netapp/trident-protect-guestfs:v2.0.0
docker.io/netapp/kopiablockrestore:26.10.1
docker.io/netapp/trident-autosupport:26.10.1
docker.io/netapp/exechook:26.10.1
docker.io/netapp/resourcebackup:26.10.1
docker.io/netapp/resourcerestore:26.10.1
docker.io/netapp/resourcedelete:26.10.1
docker.io/netapp/filebrowse-transfer:26.10.1
```

Ad esempio:

```
docker pull docker.io/netapp/controller:26.10.1
```

```
docker tag docker.io/netapp/controller:26.10.1 <private-registry-
url>/controller:26.10.1
```

```
docker push <private-registry-url>/controller:26.10.1
```

NOTA Per ottenere il grafico Helm, scarica innanzitutto il grafico Helm su una macchina con accesso a Internet utilizzando `helm pull trident-protect --version 100.2610.0 --repo https://netapp.github.io/trident-protect-helm-chart`, quindi copia il file risultante `trident-protect-100.2610.0.tgz` nel tuo ambiente offline e installalo utilizzando `helm install trident-protect ./trident-protect-100.2610.0.tgz` invece del riferimento al repository nel passaggio finale.

2. Creare lo spazio dei nomi del sistema Trident Protect:

```
kubectl create ns trident-protect
```

3. Accedere al Registro di sistema:

```
helm registry login <private-registry-url> -u <account-id> -p <api-token>
```

4. Creare un segreto pull da utilizzare per l'autenticazione privata del Registro di sistema:

```
kubectl create secret docker-registry regcred --docker-username=<registry-username> --docker-password=<api-token> -n trident-protect --docker-server=<private-registry-url>
```

5. Aggiungere il repository Trident Helm:

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

6. Crea un file denominato `protectValues.yaml` . Assicurarsi che contenga le seguenti impostazioni Trident Protect:

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

NOTA

IL `imageRegistry` E `imagePullSecrets` i valori si applicano a tutte le immagini dei componenti, comprese `resourcebackup` E `resourcerestore` . Se si inseriscono immagini in un percorso di repository specifico all'interno del registro (ad esempio, `example.com:443/my-repo`), includere il percorso completo nel campo del registro. Ciò garantirà che tutte le immagini vengano estratte da `<private-registry-url>/<image-name>:<tag>`.

7. Utilizzare Helm per installare Trident Protect. Sostituire `<name_of_cluster>` con un nome cluster, che verrà assegnato al cluster e utilizzato per identificare i backup e gli snapshot del cluster:

```
helm install trident-protect netapp-trident-protect/trident-protect --set clusterName=<name_of_cluster> --version 100.2610.0 --create-namespace --namespace trident-protect -f protectValues.yaml
```

8. Facoltativamente, per abilitare la registrazione del debug (consigliata per la risoluzione dei problemi),

utilizzare:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2610.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

La registrazione del debug aiuta NetApp a risolvere i problemi senza dover modificare il livello di registrazione o riprodurre i problemi.

NOTA Per ulteriori opzioni di configurazione del grafico Helm, incluse le impostazioni AutoSupport e il filtraggio dello spazio dei nomi, fare riferimento a "Personalizza l'installazione Trident Protect".

Installa il plugin Trident Protect CLI

È possibile utilizzare il plugin della riga di comando Trident Protect, che è un'estensione di Trident `tridentctl` utilità, per creare e interagire con le risorse personalizzate (CR) Trident Protect.

Installa il plugin Trident Protect CLI

Prima di utilizzare l'utilità della riga di comando, è necessario installarla sulla macchina utilizzata per accedere al cluster. Attenersi alla seguente procedura, a seconda che il computer utilizzi una CPU x64 o ARM.

Scarica il plugin per CPU Linux AMD64

Fasi

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-
protect/releases/download/26.10.1/tridentctl-protect-linux-amd64
```

Scarica il plugin per CPU Linux ARM64

Fasi

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-
protect/releases/download/26.10.1/tridentctl-protect-linux-arm64
```

Scarica il plugin per le CPU Mac AMD64

Fasi

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.10.1/tridentctl-protect-macos-amd64
```

Scarica il plugin per le CPU Mac ARM64

Fasi

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.10.1/tridentctl-protect-macos-arm64
```

1. Abilitare le autorizzazioni di esecuzione per il binario del plugin:

```
chmod +x tridentctl-protect
```

2. Copiare il file binario del plugin in una posizione definita nella variabile PATH. Ad esempio, `/usr/bin` o `/usr/local/bin` (potrebbe essere necessario un Privileges elevato):

```
cp ./tridentctl-protect /usr/local/bin/
```

3. Facoltativamente, è possibile copiare il file binario del plugin in una posizione nella propria home directory. In questo caso, si consiglia di assicurarsi che la posizione faccia parte della variabile PATH:

```
cp ./tridentctl-protect ~/bin/
```

NOTA Copiare il plugin in una posizione nella variabile PATH consente di utilizzare il plugin digitando `tridentctl-protect` o `tridentctl protect` da qualsiasi posizione.

Visualizza la guida del plugin CLI di Trident

È possibile utilizzare le funzioni della guida del plugin incorporato per ottenere una guida dettagliata sulle funzionalità del plugin:

Fasi

1. Utilizzare la funzione di guida per visualizzare le indicazioni sull'utilizzo:

```
tridentctl-protect help
```

Attivare il completamento automatico del comando

Dopo aver installato il plugin Trident Protect CLI, è possibile abilitare il completamento automatico per determinati comandi.

Attivare il completamento automatico per la shell Bash

Fasi

1. Creare lo script di completamento:

```
tridentctl-protect completion bash > tridentctl-completion.bash
```

2. Creare una nuova directory nella home directory in modo che contenga lo script:

```
mkdir -p ~/.bash/completions
```

3. Spostare lo script scaricato nella `~/.bash/completions` directory:

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. Aggiungere la seguente riga al `~/.bashrc` file nella propria home directory:

```
source ~/.bash/completions/tridentctl-completion.bash
```

Attivare il completamento automatico per la shell Z

Fasi

1. Creare lo script di completamento:

```
tridentctl-protect completion zsh > tridentctl-completion.zsh
```

2. Creare una nuova directory nella home directory in modo che contenga lo script:

```
mkdir -p ~/.zsh/completions
```

3. Spostare lo script scaricato nella `~/.zsh/completions` directory:

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. Aggiungere la seguente riga al `~/.zprofile` file nella propria home directory:

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

Risultato

Al prossimo login della shell, potete usare il comando auto-completation con il plugin tridentctl-Protect.

Personalizza l'installazione Trident Protect

È possibile personalizzare la configurazione predefinita di Trident Protect per soddisfare i requisiti specifici del proprio ambiente.

Specificare i limiti delle risorse del contenitore Trident Protect

Dopo aver installato Trident Protect, è possibile utilizzare un file di configurazione per specificare i limiti delle risorse per i contenitori Trident Protect. Impostando i limiti delle risorse è possibile controllare la quantità di risorse del cluster consumata dalle operazioni Trident Protect.

Fasi

1. Creare un file denominato `resourceLimits.yaml`.
2. Compilare il file con le opzioni di limitazione delle risorse per i contenitori Trident Protect in base alle esigenze del proprio ambiente.

   Il seguente file di configurazione di esempio mostra le impostazioni disponibili e contiene i valori predefiniti per ogni limite di risorse:

```
---
jobResources:
  defaults:
    limits:
      cpu: 8000m
      memory: 10000Mi
      ephemeralStorage: ""
    requests:
      cpu: 100m
      memory: 100Mi
      ephemeralStorage: ""
  resticVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  resticVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
```

```
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

3. Applicare i valori dal `resourceLimits.yaml` file:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

Personalizzare i vincoli del contesto di protezione

È possibile utilizzare un file di configurazione per modificare i vincoli di contesto di sicurezza (SCC) di OpenShift per i contenitori Trident Protect dopo aver installato Trident Protect. Questi vincoli definiscono le restrizioni di sicurezza per i pod in un cluster Red Hat OpenShift.

Fasi

1. Creare un file denominato `sccconfig.yaml`.
2. Aggiungere l'opzione SCC al file e modificare i parametri in base alle esigenze dell'ambiente.

   Nell'esempio seguente vengono mostrati i valori predefiniti dei parametri per l'opzione SCC:

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

   Questa tabella descrive i parametri per l'opzione SCC:

Parametro	Descrizione	Predefinito
creare	Determina se è possibile creare una risorsa SCC. Una risorsa SCC verrà creata solo se `scc.create` è impostato su `true` e il processo di installazione di Helm identifica un ambiente OpenShift. Se non funziona su OpenShift, o se `scc.create` è impostato su `false`, non verrà creata alcuna risorsa SCC.	vero
nome	Specifica il nome della SCC.	processo-di-protezione-Trident
priorità	Definisce la priorità dell'SCC. Gli scc con valori di priorità più elevati vengono valutati prima di quelli con valori più bassi.	1

3. Applicare i valori dal `sccconfig.yaml` file:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

In questo modo i valori predefiniti verranno sostituiti con quelli specificati nel `sccconfig.yaml` file.

Configurare le impostazioni aggiuntive del grafico del timone Trident Protect

Puoi personalizzare le impostazioni di AutoSupport, il filtro dei namespace e la gestione dei certificati webhook di OpenShift per soddisfare le tue esigenze specifiche. La tabella seguente descrive i parametri di configurazione disponibili:

Parametro	Tipo	Descrizione
autoSupport.proxy	stringa	Configura un URL proxy per le connessioni NetApp AutoSupport . Utilizzare questa opzione per instradare i caricamenti dei pacchetti di supporto tramite un server proxy. Esempio: `http://my.proxy.url` .
autoSupport.insicuro	booleano	Salta la verifica TLS per le connessioni proxy AutoSupport quando impostato su `true` . Utilizzare solo per connessioni proxy non sicure. (predefinito: `false`)

Parametro	Tipo	Descrizione
autoSupport.abilitato	booleano	Abilita o disabilita i caricamenti giornalieri del bundle Trident Protect AutoSupport . Quando impostato su `false` , i caricamenti giornalieri programmati sono disabilitati, ma puoi comunque generare manualmente i pacchetti di supporto. (predefinito: `true`)
restoreSkipNamespaceAnnotations	stringa	Elenco separato da virgole di annotazioni dello spazio dei nomi da escludere dalle operazioni di backup e ripristino. Consente di filtrare gli spazi dei nomi in base alle annotazioni.
ripristina Salta le etichette dello spazio dei nomi	stringa	Elenco separato da virgole delle etichette degli spazi dei nomi da escludere dalle operazioni di backup e ripristino. Consente di filtrare gli spazi dei nomi in base alle etichette.
useOpenshiftServiceCA	booleano	Consente l'integrazione con l'operatore CA di servizio nativo di OpenShift per la creazione e l'iniezione automatica dei certificati TLS per il webhook di ammissione di Trident Protect. Se impostato su `true`, l'operatore CA di servizio gestisce automaticamente i certificati TLS del webhook. Usa solo su cluster OpenShift. (predefinito: `false`)

Per crittografare i metadati di snapshot e backup, consulta "Crittografa i metadati di AppVault per snapshot e backup di Trident Protect".

È possibile configurare queste opzioni utilizzando un file di configurazione YAML o i flag della riga di comando:

Utilizzare il file YAML

Fasi

1. Crea un file di configurazione e assegnagli un nome `values.yaml` .
2. Nel file creato, aggiungi le opzioni di configurazione che desideri personalizzare.

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
```

```
restoreSkipNamespaceLabels: "label1,label2"
useOpenshiftServiceCA: true
```

3. Dopo aver popolato il `values.yaml` file con i valori corretti, applicare il file di configurazione:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

Usa il flag CLI

Fasi

1. Utilizzare il seguente comando con il `--set` flag per specificare parametri individuali:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set-string
restoreSkipNamespaceAnnotations="{annotation1,annotation2}" \
  --set-string restoreSkipNamespaceLabels="{label1,label2}" \
  --set useOpenshiftServiceCA=true \
  --reuse-values
```

Limita i pod Trident Protect a nodi specifici

Puoi utilizzare il vincolo di selezione dei nodi Kubernetes nodeSelector per controllare quali nodi sono idonei a eseguire i pod Trident Protect, in base alle etichette dei nodi. Per impostazione predefinita, Trident Protect è limitato ai nodi che eseguono Linux. È possibile personalizzare ulteriormente questi vincoli in base alle proprie esigenze.

Fasi

1. Creare un file denominato `nodeSelectorConfig.yaml`.
2. Aggiungere l'opzione nodeSelector al file e modificare il file per aggiungere o modificare le etichette dei nodi da limitare in base alle esigenze dell'ambiente. Ad esempio, il seguente file contiene la restrizione predefinita del sistema operativo, ma riguarda anche una regione e un nome dell'applicazione specifici:

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. Applicare i valori dal `nodeSelectorConfig.yaml` file:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

In questo modo, le restrizioni predefinite vengono sostituite da quelle specificate nel `nodeSelectorConfig.yaml` file.

Gestisci Trident Protect

Gestisci l'autorizzazione e il controllo degli accessi Trident Protect

Trident Protect utilizza il modello Kubernetes di controllo degli accessi basato sui ruoli (RBAC). Per impostazione predefinita, Trident Protect fornisce un singolo namespace di sistema e il relativo account di servizio predefinito. Se la tua organizzazione ha molti utenti o esigenze di sicurezza specifiche, puoi utilizzare le funzionalità RBAC di Trident Protect per ottenere un controllo più granulare sull'accesso alle risorse e agli spazi dei nomi.

L'amministratore del cluster ha sempre accesso alle risorse nello spazio dei nomi predefinito `trident-protect` e può anche accedere alle risorse in tutti gli altri namespace. Per controllare l'accesso a risorse e applicazioni, è necessario creare spazi dei nomi aggiuntivi e aggiungere risorse e applicazioni a tali spazi dei nomi.

Si noti che nessun utente può creare CRS per la gestione dei dati delle applicazioni nello spazio dei nomi predefinito `trident-protect`. È necessario creare CRS per la gestione dei dati delle applicazioni in uno spazio dei nomi delle applicazioni (come Best practice, creare CRS per la gestione dei dati delle applicazioni nello stesso spazio dei nomi dell'applicazione associata).

NOTA

Solo gli amministratori dovrebbero avere accesso agli oggetti di risorse personalizzate Trident Protect privilegiati, tra cui:

- **AppVault**: Richiede i dati delle credenziali del bucket
- **AutoSupportBundle**: raccoglie metriche, registri e altri dati sensibili Trident Protect
- **AutoSupportBundleSchedule**: Gestisce i programmi di raccolta dei log

Come Best practice, utilizzare RBAC per limitare l'accesso agli oggetti con privilegi agli amministratori.

Per ulteriori informazioni su come RBAC regola l'accesso alle risorse e agli spazi dei nomi, fare riferimento alla "Documentazione RBAC di Kubernetes" .

Per informazioni sugli account di servizio, fare riferimento alla "Documentazione dell'account del servizio Kubernetes" .

Esempio: Gestire l'accesso per due gruppi di utenti

Ad esempio, un'organizzazione dispone di un amministratore cluster, di un gruppo di utenti di progettazione e di un gruppo di utenti di marketing. L'amministratore del cluster dovrebbe completare le seguenti attività per creare un ambiente in cui il gruppo di progettazione e il gruppo di marketing hanno ciascuno accesso solo alle risorse assegnate ai rispettivi namespace.

Passaggio 1: Creare uno spazio dei nomi che contenga risorse per ciascun gruppo

La creazione di uno spazio dei nomi consente di separare logicamente le risorse e di controllare meglio chi ha accesso a tali risorse.

Fasi

1. Creare uno spazio dei nomi per il gruppo tecnico:

```
kubectl create ns engineering-ns
```

2. Creare uno spazio dei nomi per il gruppo di marketing:

```
kubectl create ns marketing-ns
```

Passaggio 2: Creare nuovi account di servizio per interagire con le risorse in ogni spazio dei nomi

Ogni nuovo spazio dei nomi creato viene fornito con un account di servizio predefinito, ma è necessario creare un account di servizio per ogni gruppo di utenti in modo da poter dividere ulteriormente Privileges tra i gruppi in futuro, se necessario.

Fasi

1. Creare un account di servizio per il gruppo tecnico:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. Creare un account di servizio per il gruppo di marketing:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

Passaggio 3: Creare un segreto per ogni nuovo account di servizio

Un segreto dell'account di servizio viene utilizzato per l'autenticazione con l'account di servizio e può essere facilmente eliminato e ricreato se compromesso.

Fasi

1. Creare un segreto per l'account del servizio tecnico:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: eng-user
  name: eng-user-secret
  namespace: engineering-ns
type: kubernetes.io/service-account-token
```

2. Creare un segreto per l'account del servizio di marketing:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: mkt-user
  name: mkt-user-secret
  namespace: marketing-ns
type: kubernetes.io/service-account-token
```

Passaggio 4: Creare un oggetto RoleBinding per associare l'oggetto ClusterRole a ogni nuovo account di servizio

Quando si installa Trident Protect, viene creato un oggetto ClusterRole predefinito. È possibile associare questo ClusterRole all'account di servizio creando e applicando un oggetto RoleBinding.

Fasi

1. Associare ClusterRole all'account del servizio tecnico:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: engineering-ns-tenant-rolebinding
  namespace: engineering-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

2. Associare ClusterRole all'account del servizio di marketing:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: marketing-ns-tenant-rolebinding
  namespace: marketing-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: mkt-user
  namespace: marketing-ns
```

Passaggio 5: Verifica delle autorizzazioni

Verificare che le autorizzazioni siano corrette.

Fasi

1. Verificare che gli utenti tecnici possano accedere alle risorse di progettazione:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n engineering-ns
```

2. Verificare che gli utenti tecnici non possano accedere alle risorse di marketing:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n marketing-ns
```

Passaggio 6: Concedere l'accesso agli oggetti AppVault

Per eseguire attività di gestione dei dati come backup e snapshot, l'amministratore del cluster deve garantire l'accesso agli oggetti AppVault ai singoli utenti.

Fasi

1. Creare e applicare un file YAML di combinazione di AppVault e segreto che consenta a un utente di accedere a un AppVault. Ad esempio, la seguente CR concede l'accesso ad AppVault all'utente `eng-user`:

```
apiVersion: v1
data:
  accessKeyID: <ID_value>
  secretAccessKey: <key_value>
kind: Secret
```

```
metadata:
  name: appvault-for-eng-user-only-secret
  namespace: trident-protect
type: Opaque
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: appvault-for-eng-user-only
  namespace: trident-protect # Trident Protect system namespace
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: testbucket
      endpoint: 192.168.0.1:30000
      secure: "false"
      skipCertValidation: "true"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: appvault-for-eng-user-only-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: appvault-for-eng-user-only-secret
  providerType: GenericS3
```

2. Creare e applicare un ruolo CR per consentire agli amministratori del cluster di concedere l'accesso a risorse specifiche in uno spazio dei nomi. Ad esempio:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: eng-user-appvault-reader
  namespace: trident-protect
rules:
- apiGroups:
```

```
  - protect.trident.netapp.io
  resourceNames:
  - appvault-for-enguser-only
  resources:
  - appvaults
  verbs:
  - get
```

3. Creare e applicare un RoleBinding CR per associare le autorizzazioni all'utente eng-user. Ad esempio:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eng-user-read-appvault-binding
  namespace: trident-protect
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: eng-user-appvault-reader
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

4. Verificare che le autorizzazioni siano corrette.

   a. Tentativo di recuperare le informazioni sull'oggetto AppVault per tutti gli spazi dei nomi:

   ```
   kubectl get appvaults -n trident-protect
   --as=system:serviceaccount:engineering-ns:eng-user
   ```

   L'output dovrebbe essere simile a quanto segue:

   ```
   Error from server (Forbidden): appvaults.protect.trident.netapp.io is
   forbidden: User "system:serviceaccount:engineering-ns:eng-user"
   cannot list resource "appvaults" in API group
   "protect.trident.netapp.io" in the namespace "trident-protect"
   ```

   b. Verificare se l'utente può ottenere le informazioni AppVault a cui ora dispone dell'autorizzazione per accedere:

   ```
   kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
   get appvaults.protect.trident.netapp.io/appvault-for-eng-user-only -n
   ```

```
trident-protect
```

L'output dovrebbe essere simile a quanto segue:

```
yes
```

Risultato

Gli utenti a cui sono state concesse le autorizzazioni AppVault dovrebbero essere in grado di utilizzare gli oggetti AppVault autorizzati per le operazioni di gestione dei dati delle applicazioni e non dovrebbero essere in grado di accedere a risorse esterne agli spazi dei nomi assegnati o creare nuove risorse a cui non hanno accesso.

Monitorare le risorse Trident Protect

È possibile utilizzare gli strumenti open source kube-state-metrics, Prometheus e Alertmanager per monitorare lo stato di integrità delle risorse protette da Trident Protect.

Il servizio kube-state-metrics genera metriche dalla comunicazione API di Kubernetes. Utilizzandolo con Trident Protect, puoi ottenere informazioni utili sullo stato delle risorse nel tuo ambiente.

Prometheus è un toolkit in grado di acquisire i dati generati da kube-state-metrics e presentarli come informazioni facilmente leggibili su questi oggetti. Insieme, kube-state-metrics e Prometheus ti consentono di monitorare lo stato e l'integrità delle risorse che gestisci con Trident Protect.

Alertmanager è un servizio che acquisisce gli avvisi inviati da strumenti come Prometheus e li indirizza alle destinazioni configurate dall'utente.

NOTA

Le configurazioni e le istruzioni incluse in questa procedura sono solo esempi; è necessario personalizzarle in base all'ambiente in uso. Per istruzioni specifiche e assistenza, consultare la seguente documentazione ufficiale:

- "documentazione kube-state-metrics"
- "Documentazione Prometheus"
- "Documentazione di Alertmanager"

Fase 1: Installare gli strumenti di monitoraggio

Per abilitare il monitoraggio delle risorse in Trident Protect, è necessario installare e configurare kube-state-metrics, Promethus e Alertmanager.

Installa metriche-stato-kube

È possibile installare parametri kube-state-metrics utilizzando Helm.

Fasi

1. Aggiungere il grafico Helm kube-state-metrics. Ad esempio:

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
```

```
helm repo update
```

2. Applicare il CRD di Prometheus ServiceMonitor al cluster:

```
kubectl apply -f https://raw.githubusercontent.com/prometheus-
operator/prometheus-operator/main/example/prometheus-operator-
crd/monitoring.coreos.com_servicemonitors.yaml
```

3. Creare un file di configurazione per il grafico Helm (ad esempio, `metrics-config.yaml`). È possibile personalizzare la seguente configurazione di esempio in base all'ambiente in uso:

Metrics-config.yaml: Configurazione del grafico Helm kube-state-metrics

```
---
extraArgs:
  # Collect only custom metrics
  - --custom-resource-state-only=true

customResourceState:
  enabled: true
  config:
    kind: CustomResourceStateMetrics
    spec:
      resources:
      - groupVersionKind:
          group: protect.trident.netapp.io
          kind: "Backup"
          version: "v1"
        labelsFromPath:
          backup_uid: [metadata, uid]
          backup_name: [metadata, name]
          creation_time: [metadata, creationTimestamp]
        metrics:
        - name: backup_info
          help: "Exposes details about the Backup state"
          each:
            type: Info
            info:
              labelsFromPath:
                appVaultReference: ["spec", "appVaultRef"]
                appReference: ["spec", "applicationRef"]
rbac:
  extraRules:
  - apiGroups: ["protect.trident.netapp.io"]
    resources: ["backups"]
```

```
    verbs: ["list", "watch"]

# Collect metrics from all namespaces
namespaces: ""

# Ensure that the metrics are collected by Prometheus
prometheus:
  monitor:
    enabled: true
```

4. Installare le metriche di stato kube distribuendo il grafico Helm. Ad esempio:

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. Configurare kube-state-metrics per generare metriche per le risorse personalizzate utilizzate da Trident Protect seguendo le istruzioni in "documentazione sulle risorse personalizzate kube-state-metrics" .

Installare Prometheus

È possibile installare Prometheus seguendo le istruzioni riportate nella "Documentazione Prometheus" .

Installare Alertmanager

È possibile installare Alertmanager seguendo le istruzioni riportate nella "Documentazione di Alertmanager" .

Fase 2: Configurare gli strumenti di monitoraggio per lavorare insieme

Dopo aver installato gli strumenti di monitoraggio, è necessario configurarli per lavorare insieme.

Fasi

1. Integra metriche-stato-kube con Prometheus. Modificare il file di configurazione di Prometheus (`prometheus.yaml`) e aggiungere le informazioni del servizio kube-state-metrics. Ad esempio:

   prometheus.yaml: integrazione del servizio kube-state-metrics con Prometheus

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: trident-protect
data:
  prometheus.yaml: |
    global:
      scrape_interval: 15s
    scrape_configs:
      - job_name: 'kube-state-metrics'
```

```
        static_configs:
          - targets: ['kube-state-metrics.trident-protect.svc:8080']
```

2. Configurare Prometheus per instradare gli avvisi ad Alertmanager. Modificare il file di configurazione di Prometheus (`prometheus.yaml`) e aggiungere la seguente sezione:

prometheus.yaml: Invia avvisi ad Alertmanager

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          - alertmanager.trident-protect.svc:9093
```

Risultato

Prometheus può ora raccogliere le metriche dalle metriche dello stato del kube e inviare avvisi ad Alertmanager. Ora si è pronti a configurare quali condizioni attivano un avviso e dove inviare gli avvisi.

Passaggio 3: Configurare le destinazioni degli avvisi e degli avvisi

Dopo aver configurato gli strumenti per lavorare insieme, è necessario configurare il tipo di informazioni che attivano gli avvisi e la posizione in cui devono essere inviati.

Esempio di avviso: Errore di backup

Nell'esempio seguente viene definito un avviso critico che viene attivato quando lo stato della risorsa personalizzata di backup è impostato su `Error` per 5 secondi o più. È possibile personalizzare questo esempio in base all'ambiente in uso e includere questo frammento YAML nel `prometheus.yaml` file di configurazione:

rules.yaml: Definisci un avviso Prometheus per i backup non riusciti

```
rules.yaml: |
  groups:
    - name: fail-backup
        rules:
          - alert: BackupFailed
            expr: kube_customresource_backup_info{status="Error"}
            for: 5s
            labels:
              severity: critical
            annotations:
              summary: "Backup failed"
              description: "A backup has failed."
```

Configurare Alertmanager per inviare avvisi ad altri canali

È possibile configurare Alertmanager in modo che invii notifiche ad altri canali, quali e-mail, PagerDuty,

Microsoft Teams o altri servizi di notifica specificando la rispettiva configurazione nel `alertmanager.yaml` file.

Nell'esempio seguente, Alertmanager configura l'invio di notifiche a un canale Slack. Per personalizzare questo esempio in base all'ambiente in uso, sostituire il valore della `api_url` chiave con l'URL slack webhook utilizzato nell'ambiente in uso:

alertmanager.yaml: invia avvisi a un canale Slack

```
data:
  alertmanager.yaml: |
    global:
      resolve_timeout: 5m
    route:
      receiver: 'slack-notifications'
    receivers:
      - name: 'slack-notifications'
        slack_configs:
          - api_url: '<your-slack-webhook-url>'
            channel: '#failed-backups-channel'
            send_resolved: false
```

Genera un pacchetto di supporto Trident Protect

Trident Protect consente agli amministratori di generare bundle che includono informazioni utili al supporto NetApp , tra cui registri, metriche e informazioni sulla topologia dei cluster e delle app in gestione. Se sei connesso a Internet, puoi caricare i bundle di supporto sul sito di supporto NetApp (NSS) utilizzando un file di risorse personalizzato (CR).

Creare un pacchetto di supporto utilizzando una CR

Fasi

1. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-support-bundle.yaml`).
2. Configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.triggerType**: (*required*) determina se il bundle di supporto viene generato immediatamente o pianificato. La generazione pianificata del pacchetto avviene alle 12am:00 UTC. Valori possibili:
     - Pianificato
     - Manuale
   - **Spec.uploadEnabled**: (*Optional*) Controlla se il bundle di supporto deve essere caricato nel sito di supporto NetApp dopo che è stato generato. Se non specificato, il valore predefinito è `false` . Valori possibili:

        - vero
        - false (impostazione predefinita)
    - **Spec.dataWindowStart**: (*Optional*) stringa di data in formato RFC 3339 che specifica la data e l'ora di inizio della finestra dei dati inclusi nel pacchetto di supporto. Se non specificato, il valore predefinito è 24 ore fa. La prima data della finestra che è possibile specificare è 7 giorni fa.

      Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AutoSupportBundle
metadata:
  name: trident-protect-support-bundle
spec:
  triggerType: Manual
  uploadEnabled: true
  dataWindowStart: 2024-05-05T12:30:00Z
```

3. Dopo aver popolato il `trident-protect-support-bundle.yaml` file con i valori corretti, applicare CR:

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-
protect
```

Creare un bundle di supporto utilizzando la CLI

Fasi

1. Creare il pacchetto di supporto, sostituendo i valori tra parentesi con le informazioni dell'ambiente. `trigger-type`Determina se il bundle viene creato immediatamente o se l'ora di creazione è dettata dalla pianificazione e può essere `Manual` o `Scheduled`. L'impostazione predefinita è `Manual`.

   Ad esempio:

```
tridentctl-protect create autosupportbundle <my-bundle-name>
--trigger-type <trigger-type> -n trident-protect
```

Monitorare e recuperare il pacchetto di supporto

Dopo aver creato un pacchetto di supporto utilizzando uno dei due metodi, puoi monitorarne l'avanzamento della generazione e recuperarlo nel tuo sistema locale.

Fasi

1. Aspetta il `status.generationState` raggiungere `Completed` stato. È possibile monitorare l'avanzamento della generazione con il seguente comando:

```
kubectl get autosupportbundle trident-protect-support-bundle -n trident-protect
```

2. Recupera il pacchetto di supporto sul tuo sistema locale. Ottieni il comando di copia dal bundle AutoSupport completato:

```
kubectl describe autosupportbundle trident-protect-support-bundle -n trident-protect
```

Trova il `kubectl cp` comando dall'output ed eseguilo, sostituendo l'argomento di destinazione con la directory locale preferita.

Aggiorna Trident Protect

Puoi aggiornare Trident Protect all'ultima versione per sfruttare le nuove funzionalità o le correzioni di bug.

NOTA

- Quando si esegue l'aggiornamento dalla versione 24.10, gli snapshot in esecuzione durante l'aggiornamento potrebbero non funzionare. Questo problema non impedisce la creazione di snapshot futuri, manuali o pianificati. Se uno snapshot non funziona durante l'aggiornamento, è possibile crearne manualmente uno nuovo per garantire la protezione dell'applicazione.

  Per evitare potenziali errori, è possibile disabilitare tutte le pianificazioni degli snapshot prima dell'aggiornamento e riabilitarle in seguito. Tuttavia, ciò comporterà la perdita di tutti gli snapshot pianificati durante il periodo di aggiornamento.

- Per le installazioni di registri privati, assicurati che il grafico Helm e le immagini richiesti per la versione di destinazione siano disponibili nel tuo registro privato e verifica che i tuoi valori Helm personalizzati siano compatibili con la nuova versione del grafico. Per maggiori informazioni, fare riferimento a"Installa Trident Protect da un registro privato" .

Fase 1: Selezionare una versione

Le versioni di Trident Protect seguono una convenzione di naming basata sulla data `YY.MM`, dove "YY" sono le ultime due cifre dell'anno e "MM" è il mese. Le versioni Dot seguono una convenzione `YY.MM.X`, dove "X" è il livello di patch. Selezionerai la versione a cui eseguire l'aggiornamento in base alla versione da cui stai aggiornando.

- È possibile eseguire un aggiornamento diretto a qualsiasi release di destinazione che rientri in una finestra di quattro release della versione installata. Ad esempio, è possibile eseguire un aggiornamento diretto dalla 24.10 (o da qualsiasi 24.10 dot release) alla 25.10.
- Se stai eseguendo l'aggiornamento da una versione al di fuori della finestra di quattro release, esegui un aggiornamento in più fasi. Usare le istruzioni di aggiornamento per la "versione precedente" da cui si esegue l'aggiornamento per passare alla release più recente che rientra nella finestra di quattro release. Ad esempio, se stai utilizzando la versione 24.10 e vuoi eseguire l'aggiornamento alla 26.10:
  a. Primo upgrade dal 24.10 al 25.06.
  b. Quindi esegui l'aggiornamento dalla versione 25.06 alla 26.10.

Passaggio 2: Aggiornare Trident Protect

Per aggiornare Trident Protect, procedere come segue.

Fasi

1. Aggiornare il repository di Trident Helm:

```
helm repo update
```

2. Aggiorna i CRD Trident Protect:

> **NOTA** Questo passaggio è necessario se si esegue l'aggiornamento da una versione precedente alla 25.06, poiché i CRD sono ora inclusi nella tabella Trident Protect Helm.

a. Eseguire questo comando per spostare la gestione dei CRD da `trident-protect-crds` A `trident-protect` :

```
kubectl get crd | grep protect.trident.netapp.io | awk '{print $1}' |
xargs -I {} kubectl patch crd {} --type merge -p '{"metadata":
{"annotations":{"meta.helm.sh/release-name": "trident-protect"}}}'
```

b. Esegui questo comando per eliminare il segreto Helm per `trident-protect-crds` grafico:

> **NOTA** Non disinstallare il `trident-protect-crds` grafico utilizzando Helm, poiché ciò potrebbe rimuovere i CRD e tutti i dati correlati.

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. Aggiorna Trident Protect:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2610.0 --namespace trident-protect
```

> **NOTA** È possibile configurare il livello di registrazione durante l'aggiornamento aggiungendo `--set logLevel=debug` al comando di aggiornamento. Il livello di registrazione predefinito è `warn` . La registrazione del debug è consigliata per la risoluzione dei problemi, poiché aiuta il supporto NetApp a diagnosticare i problemi senza richiedere modifiche al livello di registro o la riproduzione del problema.

Gestisci e proteggi le applicazioni

Utilizzare gli oggetti Trident Protect AppVault per gestire i bucket

La risorsa personalizzata del bucket (CR) per Trident Protect è nota come AppVault. Gli oggetti AppVault sono la rappresentazione dichiarativa del flusso di lavoro Kubernetes di un bucket di archiviazione. Un CR di AppVault contiene le configurazioni necessarie affinché un bucket venga utilizzato nelle operazioni di protezione, come backup, snapshot, operazioni di ripristino e replica SnapMirror . Solo gli amministratori possono creare AppVault.

Quando si eseguono operazioni di protezione dei dati su un'applicazione, è necessario creare una CR di AppVault manualmente o dalla riga di comando. La CR di AppVault è specifica per il proprio ambiente e gli esempi in questa pagina possono essere utilizzati come guida per la creazione di CR di AppVault.

NOTA Assicurarsi che AppVault CR si trovi nel cluster in cui è installato Trident Protect. Se la CR di AppVault non esiste o non è possibile accedervi, la riga di comando mostrerà un errore.

Configurare l'autenticazione e le password AppVault

Prima di creare una CR di AppVault, assicurati che AppVault e il data mover scelto possano autenticarsi con il provider e con tutte le risorse correlate.

Password del repository di spostamento dati

Quando si creano oggetti AppVault utilizzando CR o il plug-in Trident Protect CLI, è possibile specificare un segreto Kubernetes con password personalizzate per la crittografia Restic e Kopia. Se non specifichi un segreto, Trident Protect utilizza una password predefinita.

- Quando si creano manualmente CR di AppVault, utilizzare il campo **spec.dataMoverPasswordSecretRef** per specificare il segreto.
- Quando si creano oggetti AppVault utilizzando la CLI Trident Protect, utilizzare `--data-mover -password-secret-ref` argomento per specificare il segreto.

NOTA Per crittografare i metadati di snapshot e backup o per modificare la password del vault, utilizza il campo `spec.vaultPasswordSecretRef`. Consulta "Crittografa i metadati di AppVault per snapshot e backup di Trident Protect".

Creare una password segreta dell'archivio di spostamento dati

Utilizzare i seguenti esempi per creare la password segreta. Quando si creano oggetti AppVault, è possibile indicare a Trident Protect di utilizzare questo segreto per l'autenticazione con il repository del data mover.

NOTA

- A seconda di quale strumento di spostamento dati si sta utilizzando, è sufficiente includere la password corrispondente per tale strumento. Ad esempio, se si sta utilizzando Restic e non si prevede di utilizzare Kopia in futuro, è possibile includere solo la password Restic quando si crea il segreto.
- Conservare la password in un luogo sicuro. Sarà necessaria per ripristinare i dati sullo stesso cluster o su uno diverso. Se il cluster o il `trident-protect` Se lo spazio dei nomi viene eliminato, non sarà possibile ripristinare i backup o gli snapshot senza la password.

Utilizzare un CR

```
---
apiVersion: v1
data:
  KOPIA_PASSWORD: <base64-encoded-password>
  RESTIC_PASSWORD: <base64-encoded-password>
kind: Secret
metadata:
  name: my-optional-data-mover-secret
  namespace: trident-protect
type: Opaque
```

Utilizzare la CLI

```
kubectl create secret generic my-optional-data-mover-secret \
--from-literal=KOPIA_PASSWORD=<plain-text-password> \
--from-literal=RESTIC_PASSWORD=<plain-text-password> \
-n trident-protect
```

Autorizzazioni IAM per l'archiviazione compatibile con S3

Quando si accede a un archivio compatibile con S3 come Amazon S3, Generic S3, "StorageGRID S3" , O "ONTAP S3" utilizzando Trident Protect, è necessario assicurarsi che le credenziali utente fornite dispongano delle autorizzazioni necessarie per accedere al bucket. Di seguito è riportato un esempio di policy che concede le autorizzazioni minime richieste per l'accesso con Trident Protect. È possibile applicare questo criterio all'utente che gestisce i criteri dei bucket compatibili con S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

Per ulteriori informazioni sulle policy di Amazon S3, fare riferimento agli esempi in "Documentazione di Amazon S3" .

Identità pod EKS per l'autenticazione Amazon S3 (AWS)

Trident Protect supporta EKS Pod Identity per le operazioni di spostamento dati Kopia. Questa funzionalità consente l'accesso sicuro ai bucket S3 senza dover archiviare le credenziali AWS nei segreti di Kubernetes.

Requisiti per l'identità del pod EKS con Trident Protect

Prima di utilizzare EKS Pod Identity con Trident Protect, accertarsi di quanto segue:

- Il tuo cluster EKS ha l'identità Pod abilitata.
- Hai creato un ruolo IAM con le autorizzazioni necessarie per il bucket S3. Per saperne di più, fare riferimento a"Autorizzazioni IAM per l'archiviazione compatibile con S3" .
- Il ruolo IAM è associato ai seguenti account di servizio Trident Protect:
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

Per istruzioni dettagliate sull'abilitazione di Pod Identity e sull'associazione dei ruoli IAM agli account di servizio, fare riferimento a "Documentazione sull'identità del pod AWS EKS" .

Configurazione AppVault Quando si utilizza EKS Pod Identity, configurare il CR AppVault con `useIAM: true` flag invece di credenziali esplicite:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: eks-protect-vault
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-aws
      endpoint: s3.example.com
      useIAM: true
```

Esempi di generazione delle chiavi AppVault per i cloud provider

Quando si definisce un CR AppVault, è necessario includere le credenziali per accedere alle risorse ospitate dal provider, a meno che non si utilizzi l'autenticazione IAM. Il modo in cui vengono generate le chiavi per le credenziali varia a seconda del provider. Di seguito sono riportati esempi di generazione di chiavi da riga di comando per diversi provider. È possibile utilizzare gli esempi seguenti per creare chiavi per le credenziali di ciascun provider cloud.

Google Cloud

```
kubectl create secret generic <secret-name> \
--from-file=credentials=<mycreds-file.json> \
-n trident-protect
```

Amazon S3 (AWS)

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<amazon-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

Microsoft Azure

```
kubectl create secret generic <secret-name> \
--from-literal=accountKey=<secret-name> \
-n trident-protect
```

Generico S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<generic-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

ONTAP S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<ontap-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

StorageGRID S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<storagegrid-s3-trident-protect-src
-bucket-secret> \
-n trident-protect
```

Esempi di creazione di AppVault

Di seguito sono riportate alcune definizioni AppVault di esempio per ogni provider.

Esempi di AppVault CR

È possibile utilizzare i seguenti esempi CR per creare oggetti AppVault per ciascun provider cloud.

NOTA

- Puoi anche specificare un Kubernetes Secret che contiene password personalizzate per la crittografia dei repository Restic e Kopia. Per ulteriori informazioni, fare riferimento Password del repository di spostamento dati a.
- Per gli oggetti AppVault di Amazon S3 (AWS), è possibile specificare un oggetto sessionToken, utile se si utilizza il Single Sign-on (SSO) per l'autenticazione. Questo token viene creato quando si generano le chiavi per il provider in Esempi di generazione delle chiavi AppVault per i cloud provider.
- Per gli oggetti AppVault S3, è possibile specificare facoltativamente un URL proxy di uscita per il traffico S3 in uscita utilizzando la `spec.providerConfig.S3.proxyURL` chiave.

Google Cloud

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

Amazon S3 (AWS)

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: amazon-s3-trident-protect-src-bucket
  namespace: trident-protect
```

```
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
    sessionToken:
      valueFromSecret:
        key: sessionToken
        name: s3-secret
```

NOTA Per gli ambienti EKS che utilizzano Pod Identity con Kopia Data Mover, è possibile rimuovere `providerCredentials` sezione e aggiungi `useIAM: true` sotto il `s3` configurazione invece.

Microsoft Azure

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

Generico S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: generic-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

ONTAP S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: OntapS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
```

```
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

StorageGRID S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

Esempi di creazione di AppVault utilizzando la CLI Trident Protect

È possibile utilizzare i seguenti esempi di comandi CLI per creare CRS AppVault per ciascun provider.

NOTA

- Puoi anche specificare un Kubernetes Secret che contiene password personalizzate per la crittografia dei repository Restic e Kopia. Per ulteriori informazioni, fare riferimento Password del repository di spostamento dati a.
- Per gli oggetti AppVault S3, è possibile specificare facoltativamente un URL proxy di uscita per il traffico S3 in uscita utilizzando l' `--proxy-url <ip_address:port>`argomento.

Google Cloud

```
tridentctl-protect create vault GCP <vault-name> \
--bucket <mybucket> \
--project <my-gcp-project> \
--secret <secret-name>/credentials \
```

```
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Amazon S3 (AWS)

```
tridentctl-protect create vault AWS <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Microsoft Azure

```
tridentctl-protect create vault Azure <vault-name> \
--account <account-name> \
--bucket <bucket-name> \
--secret <secret-name> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Generico S3

```
tridentctl-protect create vault GenericS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

ONTAP S3

```
tridentctl-protect create vault OntapS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

StorageGRID S3

```
tridentctl-protect create vault StorageGridS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
```

```
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Supportato `providerConfig.s3` opzioni di configurazione

Per le opzioni di configurazione del provider S3, consultare la seguente tabella:

Parametro	Descrizione	Predefinito	Esempio
`providerConfig.s3.skipCertValidation`	Disattiva la verifica del certificato SSL/TLS.	falso	"vero", "falso"
`providerConfig.s3.secure`	Abilita la comunicazione HTTPS sicura con l'endpoint S3.	vero	"vero", "falso"
`providerConfig.s3.proxyURL`	Specificare l'URL del server proxy utilizzato per connettersi a S3.	Nessuno	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	Fornire un certificato CA radice personalizzato per la verifica SSL/TLS.	Nessuno	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	Abilita l'autenticazione IAM per accedere ai bucket S3. Applicabile per l'identità del pod EKS.	falso	vero, falso

Visualizzare le informazioni AppVault

È possibile utilizzare il plug-in Trident Protect CLI per visualizzare informazioni sugli oggetti AppVault creati nel cluster.

Fasi

1. Visualizzare il contenuto di un oggetto AppVault:

```
tridentctl-protect get appvaultcontent gcp-vault \
--show-resources all \
-n trident-protect
```

Output di esempio:

```
+-------------+-------+----------+----------------------------
+--------------------------+
|   CLUSTER   |  APP  |   TYPE   |            NAME             |
TIMESTAMP         |
```

```
+-------------+-------+----------+----------------------------
+--------------------------+
|             | mysql | snapshot | mysnap                       | 2024-
08-09 21:02:11 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815180300 | 2024-
08-15 18:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815200300 | 2024-
08-15 20:03:06 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815180300 | 2024-
08-15 18:04:25 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:30 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815200300 | 2024-
08-15 20:04:21 (UTC) |
| production1 | mysql | backup   | mybackup5                    | 2024-
08-09 22:25:13 (UTC) |
|             | mysql | backup   | mybackup                     | 2024-
08-09 21:02:52 (UTC) |
+-------------+-------+----------+----------------------------
+--------------------------+
```

2. Facoltativamente, per visualizzare AppVaultPath per ogni risorsa, utilizzare il flag `--show-paths`.

   Il nome del cluster nella prima colonna della tabella è disponibile solo se è stato specificato un nome del cluster nell'installazione di Trident Protect Helm. Per esempio: `--set clusterName=production1` .

Rimuovere un AppVault

È possibile rimuovere un oggetto AppVault in qualsiasi momento.

NOTA	Non rimuovere la `finalizers` chiave in AppVault CR prima di eliminare l'oggetto AppVault. In tal caso, i dati residui nel bucket AppVault e le risorse orfane nel cluster possono risultare.

Prima di iniziare

Assicurarsi di aver eliminato tutti i CRS di backup e snapshot utilizzati dall'AppVault che si desidera eliminare.

Rimuovere un AppVault usando l'interfaccia a riga di comando di Kubernetes

1. Rimuovere l'oggetto AppVault, sostituendo `appvault-name` con il nome dell'oggetto AppVault da rimuovere:

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

Rimuovere un AppVault utilizzando la CLI Trident Protect

1. Rimuovere l'oggetto AppVault, sostituendo `appvault-name` con il nome dell'oggetto AppVault da rimuovere:

```
tridentctl-protect delete appvault <appvault-name> \
-n trident-protect
```

Crittografa i metadati di AppVault per snapshot e backup di Trident Protect

Puoi crittografare i metadati di snapshot e backup che Trident Protect memorizza in un AppVault, incluse le informazioni sensibili nei Kubernetes Secrets. Puoi anche modificare la password che protegge i dati crittografati. Una singola modifica della password crittografa nuovamente la chiave di crittografia dei metadati e sposta ogni repository Kopia e ogni repository Restic nell'AppVault alla nuova password.

La crittografia è disabilitata per impostazione predefinita. Quando la abiliti, Trident Protect crittografa i nuovi metadati di snapshot e backup prima di archiviarli in AppVault. Questo protegge le risorse dell'applicazione, incluse le informazioni sensibili in Secrets, a meno che tu non escluda esplicitamente tali risorse dallo snapshot o dal backup dell'applicazione.

Una password del vault protegge la chiave che crittografa i dati. Tieni presente il seguente comportamento:

- Quando modifichi la password del vault, Trident Protect protegge la stessa chiave con la nuova password. Non sovrascrive i dati già crittografati.
- I dati memorizzati prima di attivare la crittografia rimangono leggibili.
- I dati crittografati rimangono leggibili, anche se in seguito disattivi la crittografia per i nuovi dati.

A proposito di questa attività

Quando attivi la crittografia, Trident Protect utilizza una password predefinita non sicura a meno che non specifichi una password personalizzata. Per ogni AppVault, crea una password del vault personalizzata e fai riferimento a essa in `spec.vaultPasswordSecretRef`.

Se hai abilitato la crittografia con la password predefinita e in seguito hai aggiunto una password personalizzata, devi ruotare la password in modo che i dati e i repository esistenti passino alla password personalizzata. Vedi Ruota la password del vault.

Utilizza una password personalizzata univoca per ciascun AppVault. Ruota la password di un solo AppVault alla volta e verifica che la rotazione sia completata prima di iniziare un'altra modifica della password.

Prima di iniziare

Assicurati di avere quanto segue:

- Accesso allo `trident-protect` spazio dei nomi.
- Autorizzazione a creare Secrets di Kubernetes e applicare patch alle risorse personalizzate (CR) di AppVault.
- È prevista una password segreta separata per ciascun AppVault.

Abilita la crittografia dei metadati

Abiliti la crittografia dei metadati impostando un valore Helm per Trident Protect.

Fasi

1. Imposta `resourceBackupEncryptionEnabled` su `true` usando un `values.yaml` file:
   a. Aggiungi quanto segue al tuo `values.yaml` file:

   ```
   resourceBackupEncryptionEnabled: true
   ```

   b. Applica il file:

   ```
   helm upgrade trident-protect -n trident-protect netapp-trident-
   protect/trident-protect -f values.yaml --reuse-values
   ```

2. In alternativa, imposta `resourceBackupEncryptionEnabled` su `true` usando il flag Helm `--set`:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set resourceBackupEncryptionEnabled=true \
  --reuse-values
```

Risultato

Trident Protect crittografa i nuovi metadati di snapshot e backup.

Quali sono le prossime novità?

Crea una password del vault personalizzata per ogni AppVault. Finché non lo fai, Trident Protect usa la password predefinita non sicura.

Crea una password personalizzata per il vault

Per usare la tua password, crea un Kubernetes Secret che contenga la password, quindi fai riferimento al Secret nell'AppVault.

A proposito di questa attività

Se configuri la password personalizzata quando crei AppVault, prima che vengano memorizzati dati, AppVault utilizza immediatamente la password personalizzata. Non devi ruotare la password.

Se AppVault contiene già repository Kopia o Restic, l'aggiunta di una nuova password non modifica la password di tali repository. Non usare i passaggi in questa sezione. Segui invece i passaggi in Ruota la password del vault.

Il `VAULT_PASSWORD` valore deve soddisfare i seguenti requisiti:

- Non deve essere vuoto
- Deve essere più breve di 64 KiB

- Non deve contenere il carattere NUL
- Non deve contenere un'interruzione di riga
- Non deve iniziare o terminare con uno spazio o altro carattere vuoto

La password può contenere altri caratteri, inclusi caratteri non ASCII.

Fasi

1. Crea un Secret nello stesso namespace di AppVault. Il Secret deve contenere una chiave denominata `VAULT_PASSWORD`:

```
kubectl -n trident-protect create secret generic vault-password \
  --from-literal=VAULT_PASSWORD='<password>'
```

> **NOTA** Se AppVault segnala che la password non è valida, verifica il `VAULT_PASSWORD` valore rispetto ai requisiti sopra indicati. Le cause più comuni sono un valore vuoto, un carattere di fine riga finale o spazi all'inizio o alla fine della password.

2. Fai riferimento al segreto nel campo `spec.vaultPasswordSecretRef` della CR AppVault:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: my-vault
  namespace: trident-protect
spec:
  vaultPasswordSecretRef: vault-password
  providerType: <provider-type>
  providerConfig:
    <provider-configuration>
```

   Se `spec.vaultPasswordSecretRef` è vuoto, AppVault usa la password predefinita non sicura.

Risultato

AppVault utilizza la tua password personalizzata per proteggere la chiave di crittografia dei metadati.

Ruota la password del vault

Puoi ruotare la password per un AppVault che archivia già dati. Una rotazione crittografa di nuovo la chiave di crittografia dei metadati e sposta ogni repository Kopia e ogni repository Restic nel AppVault alla nuova password.

A proposito di questa attività

Durante la rotazione, Trident Protect esegue le seguenti operazioni:

- Attende il completamento dei processi di backup e ripristino in esecuzione per AppVault prima di avviarsi.
- Protegge la chiave di crittografia dei metadati (`metadata-key.json`) con la nuova password. Non

riscrive gli oggetti dell'archivio.

- Trova ogni repository Kopia e ogni repository Restic con il prefisso AppVault e ne cambia la password con la nuova password. I repository che si aprono già con la nuova password non vengono modificati.
- Salta i repository che un processo di backup o ripristino sta ancora utilizzando e li riprova più tardi.
- Sospende le pianificazioni abilitate che fanno riferimento a AppVault. Per ogni pianificazione, Trident Protect imposta `spec.enabled` su `false` e aggiunge la seguente annotazione:

```
protect.trident.netapp.io/vault-password-rotation-suspended:
<namespace>/<appvault-name>
```

Al termine della rotazione, Trident Protect riattiva solo le pianificazioni che presentano questa annotazione. In caso di errore durante la rotazione, questa riprende dal punto in cui si era interrotta. I repository che utilizzano già la nuova password non vengono modificati nuovamente.

Prima di iniziare

Assicurati di avere quanto segue:

- Una nuova password che soddisfi requisiti della password del vault.
- Nessun altro AppVault sta ruotando la propria password. Se più AppVaults condividono lo stesso bucket e prefisso, solo uno di essi può ruotare la password. Consulta Gestisci la rotazione per AppVaults che condividono un bucket.

Fasi

1. Crea un nuovo Secret con un nome diverso da quello del Secret attuale:

```
kubectl -n trident-protect create secret generic vault-password-new \
  --from-literal=VAULT_PASSWORD='<new-password>'
```

IMPORTANTE	Non modificare o eliminare il Secret corrente prima del completamento della rotazione. Trident Protect necessita della password corrente per accedere ai dati esistenti finché tutti i repository non utilizzano la nuova password. Se elimini il Secret corrente troppo presto, ricrealo con lo stesso nome e il valore originale `VAULT_PASSWORD`.

2. Abilita la rotazione e imposta AppVault in modo che utilizzi il nuovo Secret. Sostituisci `my-vault` con il nome del tuo AppVault:

```
kubectl -n trident-protect patch appvault my-vault --type merge --patch
'
spec:
  enableRotation: true
  vaultPasswordSecretRef: vault-password-new
'
```

NOTA	Devi impostare `spec.enableRotation` su `true` prima di cambiare la password. Se è `false`, AppVault segnala che la rotazione è disabilitata e non cambia la password.

3. Monitora la rotazione seguendo i passaggi descritti in Monitora la rotazione delle password.
4. Al termine della rotazione, elimina il Secret precedente se nessun altro lo utilizza.

Risultato

La chiave di crittografia dei metadati e ogni repository Kopia e Restic nell'AppVault sono protetti dalla nuova password e le pianificazioni che Trident Protect aveva sospeso sono di nuovo abilitate.

Monitora la rotazione delle password

Puoi verificare l'avanzamento della rotazione delle password e confermare che è completa.

Fasi

1. Visualizza AppVault:

```
kubectl -n trident-protect get appvault my-vault --output yaml
```

2. Nell'output, controlla la `status.vaultPasswordRotation` sezione. Durante la rotazione, ha questo aspetto:

```
status:
  vaultPasswordRotation:
    state: Rotating
    rotated: 3
    pending: 1
    message: <message>
```

Questa tabella descrive i campi:

Campo	Descrizione
status.vaultPasswordRotation.state	Stato di rotazione attuale.
stato.vaultPasswordRotation.ruotato	Numero di repository che ora utilizzano la nuova password.
stato.vaultPasswordRotation.in sospeso	Numero di repository ignorati perché in uso da un backup o da un ripristino.
stato.vaultPasswordRotation.messaggio	Ultimo messaggio di avanzamento o di errore.

Risultato

La rotazione è completa quando sono vere tutte le seguenti condizioni:

- `status.vaultPasswordRotation` non è presente.
- `status.previousRepositoryPasswordSecretRef` è vuoto.

- `status.repositoryPasswordSecretRef`corrispondenze `spec.vaultPasswordSecretRef`.

Quali sono le prossime novità?

Se la rotazione non termina, controlla quanto segue:

- **La rotazione non si avvia**: `spec.enableRotation` potrebbe essere `false`. Impostalo su `true` e poi imposta `spec.vaultPasswordSecretRef` sul nuovo Secret.
- **I repository rimangono in sospeso**: i processi di backup o ripristino li stanno ancora utilizzando. Attendi il completamento dei processi. Trident Protect riprova automaticamente i repository in sospeso.
- **La rotazione non si avvia perché un altro AppVault possiede il lock**: fai riferimento a Gestisci la rotazione per AppVaults che condividono un bucket.
- **Il campo messaggio segnala un errore**: Risolvi il problema descritto nel messaggio. La rotazione riprende da dove si era interrotta.

Migra da dataMoverPasswordSecretRef

Se un AppVault esistente utilizza il campo `spec.dataMoverPasswordSecretRef` precedente, puoi spostarlo in `spec.vaultPasswordSecretRef`.

A proposito di questa attività

Trident Protect mantiene `spec.dataMoverPasswordSecretRef` per la retrocompatibilità. Usa `spec.vaultPasswordSecretRef` per i nuovi AppVaults.

Fasi

1. Mantieni `spec.dataMoverPasswordSecretRef` e il relativo Secret in posizione.
2. Ruota la password in un nuovo Secret utilizzando `spec.vaultPasswordSecretRef`. Segui i passaggi in Ruota la password del vault.
3. Al termine della rotazione, rimuovi `spec.dataMoverPasswordSecretRef` da AppVault.
4. Elimina il vecchio Secret se nient'altro lo usa.

Gestisci la rotazione per AppVaults che condividono un bucket

Se diversi AppVaults utilizzano lo stesso bucket e prefisso, solo uno di essi può ruotare la password.

A proposito di questa attività

La rotazione delle password utilizza un file di blocco denominato `password-rotation.lock`. Il blocco registra l'UID di AppVault autorizzato a ruotare la password per il bucket e il prefisso. Se un secondo AppVault ha `spec.enableRotation: true` e il blocco indica un AppVault diverso, il secondo AppVault non può eseguire la rotazione.

Esempio 1. Fasi

Trasferisci la rotazione a un altro AppVault

1. Su AppVault che possiede il blocco, cancella `spec.enableRotation`:

```
kubectl -n trident-protect patch appvault <current-owner> --type
merge --patch '
spec:
```

```
  enableRotation: false
'
```

2. Attendi che il blocco venga rimosso.
3. Sullo AppVault che dovrebbe ruotare successivamente, imposta `spec.enableRotation` su `true`:

```
kubectl -n trident-protect patch appvault <new-owner> --type merge
--patch '
spec:
  enableRotation: true
'
```

Rimuovi un blocco lasciato da AppVault eliminato

Segui questi passaggi se AppVault, che possiede il blocco, non esiste più e il blocco non è stato rimosso, ad esempio dopo un arresto anomalo.

1. Elimina il `password-rotation.lock` file dal prefisso AppVault nel tuo bucket.
2. Su AppVault che dovrebbe ruotare, imposta `spec.enableRotation` su `true`:

```
kubectl -n trident-protect patch appvault <appvault-name> --type
merge --patch '
spec:
  enableRotation: true
'
```

Informazioni correlate

- "Utilizzare gli oggetti Trident Protect AppVault per gestire i bucket"
- "Personalizza l'installazione Trident Protect"
- "Proteggere le applicazioni"
- "Ripristina le applicazioni utilizzando Trident Protect"

Definisci un'applicazione per la gestione con Trident Protect

È possibile definire un'applicazione che si desidera gestire con Trident Protect creando una CR dell'applicazione e una CR AppVault associata.

Creare un AppVault CR

È necessario creare un CR AppVault che verrà utilizzato durante l'esecuzione di operazioni di protezione dei dati sull'applicazione e il CR AppVault deve risiedere nel cluster in cui è installato Trident Protect. Il CR di AppVault è specifico per il tuo ambiente; per esempi di CR di AppVault, fai riferimento a"Risorse personalizzate AppVault."

Definire un'applicazione

È necessario definire ciascuna applicazione che si desidera gestire con Trident Protect. È possibile definire un'applicazione per la gestione creando manualmente un CR dell'applicazione oppure utilizzando la CLI Trident Protect.

NOTA Per definire un'applicazione con ambito VM che protegge specifiche KubeVirt macchine virtuali (VM) per nome anziché interi namespace, usa il campo `spec.includedVirtualMachines` nell'Application CR o il flag `--virtual-machines` nella CLI. Per dettagli, riferimenti ai campi ed esempi YAML, consulta "Proteggi le macchine virtuali KubeVirt con Trident Protect".

Aggiungere un'applicazione utilizzando una CR

Fasi

1. Creare il file CR dell'applicazione di destinazione:
   a. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `maria-app.yaml`).
   b. Configurare i seguenti attributi:
      - **metadata.name**: (*required*) il nome della risorsa personalizzata dell'applicazione. Si noti il nome scelto perché altri file CR necessari per le operazioni di protezione fanno riferimento a questo valore.
      - **spec.includedNamespaces**: (*required*) utilizzare lo spazio dei nomi e il selettore di etichette per specificare gli spazi dei nomi e le risorse utilizzate dall'applicazione. Lo spazio dei nomi dell'applicazione deve far parte di questo elenco. Il selettore delle etichette è opzionale e può essere utilizzato per filtrare le risorse all'interno di ogni spazio dei nomi specificato.
      - **spec.includedClusterScopedResources**: (*Optional*) utilizzare questo attributo per specificare le risorse con ambito cluster da includere nella definizione dell'applicazione. Questo attributo consente di selezionare queste risorse in base al gruppo, alla versione, al tipo e alle etichette.
        - **GroupVersionKind**: (*required*) specifica il gruppo API, la versione e il tipo di risorsa con ambito cluster.
        - **LabelSelector**: (*Optional*) Filtra le risorse con ambito cluster in base alle loro etichette.
      - **spec.volumeGroups**: (*Optional*) Usa questo attributo per creare snapshot insieme di volumi correlati come snapshot di gruppi di volumi coerenti in caso di crash. Per i dettagli, consulta "Utilizza gli snapshot dei gruppi di volumi con Trident Protect".
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Facoltativo*) Questa annotazione è applicabile solo alle applicazioni definite da macchine virtuali, come negli ambienti KubeVirt, in cui i blocchi del file system si verificano prima degli snapshot. Specificare se questa applicazione può scrivere sul file system durante uno snapshot. Se impostato su true, l'applicazione ignora l'impostazione globale e può scrivere sul file system durante uno snapshot. Se impostato su false, l'applicazione ignora l'impostazione globale e il file system viene bloccato durante uno snapshot. Se specificato ma l'applicazione non ha macchine virtuali nella definizione dell'applicazione, l'annotazione viene ignorata. Se non specificato, l'applicazione segue la"impostazione di congelamento globale Trident Protect" .

Se è necessario applicare questa annotazione dopo la creazione di un'applicazione, è possibile utilizzare il seguente comando:

```
kubectl annotate application -n <application CR namespace> <application CR name> protect.trident.netapp.io/skip-vm-freeze="true"
```

+
Esempio YAML:

+

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  annotations:
    protect.trident.netapp.io/skip-vm-freeze: "false"
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: namespace-1
      labelSelector:
        matchLabels:
          app: example-app
    - namespace: namespace-2
      labelSelector:
        matchLabels:
          app: another-example-app
  includedClusterScopedResources:
    - groupVersionKind:
        group: rbac.authorization.k8s.io
        kind: ClusterRole
        version: v1
      labelSelector:
        matchLabels:
          mylabel: test
```

1. (*Facoltativo*) Se necessario, puoi aggiungere il filtro delle risorse alla stessa CR per includere o escludere risorse specifiche:

   - **Esempio di filtro generico:**

     - **ResourceFilter.resourceSelectionCriteria**: (Necessario per il filtraggio) utilizzare `Include` o includere o `Exclude` escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti parametri resourceMatcher per definire le risorse da includere o escludere:

       - **ResourceFilter.resourceMatchers**: Una matrice di oggetti resourceMatcher. Se si definiscono più elementi in questa matrice, questi corrispondono come un'operazione OR

e i campi all'interno di ogni elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.

- **ResourceMatchers[].group**: (*Optional*) Gruppo della risorsa da filtrare.
- **ResourceMatchers[].Kind**: (*Optional*) tipo di risorsa da filtrare.
- **ResourceMatchers[].version**: (*Optional*) versione della risorsa da filtrare.
- **ResourceMatchers[].names**: (*Optional*) nomi nel campo Kubernetes metadata.name della risorsa da filtrare.
- **ResourceMatchers[].namespaces**: (*Optional*) Namespaces nel campo Kubernetes metadata.name della risorsa da filtrare.
- **ResourceMatchers[].labelSelectors**: (*Optional*) stringa del selettore di etichette nel campo Kubernetes metadata.name della risorsa come definito nella "Documentazione Kubernetes" . Ad esempio: `"trident.netapp.io/os=linux"`.

NOTA	Quando entrambi `resourceFilter` E `labelSelector` vengono utilizzati, `resourceFilter` corre prima e poi `labelSelector` viene applicato alle risorse risultanti.

Ad esempio:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

◦ **Esempio di filtro solo PVC:**

Per definire un'applicazione solo PVC, è necessario includere anche `PersistentVolume` e `VolumeSnapshotClass` nel filtro delle risorse. Le operazioni di Snapshot e backup dipendono da `PersistentVolume` (il volume con ambito cluster associato a ciascun PVC) e `VolumeSnapshotClass` (il driver di snapshot), e non funzioneranno senza di essi. Ad esempio:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
```

```
metadata:
  name: my-pvc-app
  namespace: my-app-namespace
spec:
  includedNamespaces:
  - namespace: my-app-namespace
  resourceFilter:
    resourceMatchers:
    - kind: PersistentVolumeClaim
      version: v1
    - kind: PersistentVolume
      version: v1
    - kind: VolumeSnapshotClass
      version: v1
    resourceSelectionCriteria: Include
```

2. Dopo aver creato l'applicazione CR per adattarla all'ambiente in uso, applicare il CR. Ad esempio:

```
kubectl apply -f maria-app.yaml
```

Fasi

1. Creare e applicare la definizione dell'applicazione utilizzando uno dei seguenti esempi, sostituendo i valori tra parentesi con le informazioni dell'ambiente. È possibile includere spazi dei nomi e risorse nella definizione dell'applicazione utilizzando elenchi separati da virgole con gli argomenti illustrati negli esempi.

Facoltativamente, quando si crea un'app è possibile utilizzare un'annotazione per specificare se l'applicazione può scrivere sul file system durante uno snapshot. Ciò è applicabile solo alle applicazioni definite da macchine virtuali, come negli ambienti KubeVirt, in cui si verificano blocchi del file system prima degli snapshot. Se si imposta l'annotazione su `true` , l'applicazione ignora l'impostazione globale e può scrivere sul file system durante uno snapshot. Se lo imposti su `false` , l'applicazione ignora l'impostazione globale e il file system viene bloccato durante uno snapshot. Se si utilizza l'annotazione ma l'applicazione non ha macchine virtuali nella definizione dell'applicazione, l'annotazione viene ignorata. Se non si utilizza l'annotazione, l'applicazione segue la"impostazione di congelamento globale Trident Protect" .

+ Per specificare l'annotazione quando usi la CLI per creare un'applicazione, puoi usare il `--annotation` flag.

+ * Crea l'applicazione e utilizza l'impostazione globale per il comportamento di blocco del filesystem:

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace>
```

+ * Crea l'applicazione e configura le impostazioni locali dell'applicazione per il comportamento di blocco del filesystem:

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace> --annotation
protect.trident.netapp.io/skip-vm-freeze=<"true"|"false">
```

Puoi usare `--resource-filter-include` e `--resource-filter-exclude` flag per includere o escludere risorse in base a `resourceSelectionCriteria` come gruppo, tipo, versione, etichette, nomi e namespace, come mostrato nel seguente esempio:

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace> --resource-filter-include
'[{"Group":"apps","Kind":"Deployment","Version":"v1","Names":["my-
deployment"],"Namespaces":["my-namespace"],"LabelSelectors":["app=my-
app"]}]'
```

+ * Per definire un'applicazione solo PVC, devi includere anche `PersistentVolume` e `VolumeSnapshotClass` nel filtro delle risorse. Le operazioni di Snapshot e backup dipendono da `PersistentVolume` (il volume con ambito cluster associato a ciascun PVC) e `VolumeSnapshotClass` (il driver di Snapshot), e non funzioneranno senza di essi. Per esempio:

+

```
tridentctl-protect create app my-pvc-app --namespaces <my-app-
namespace> --resource-filter-include
'[{"Kind":"PersistentVolumeClaim","Version":"v1"},{"Kind":"PersistentVo
lume","Version":"v1"},{"Kind":"VolumeSnapshotClass","Version":"v1"}]'
-n <my-app-namespace>
```

Proteggi le applicazioni utilizzando Trident Protect

È possibile proteggere tutte le app gestite da Trident Protect eseguendo snapshot e backup tramite una policy di protezione automatizzata o su base ad hoc.

NOTA

- È possibile configurare Trident Protect in modo che blocchi e sblocchi i file system durante le operazioni di protezione dei dati. "Scopri di più sulla configurazione del congelamento del file system con Trident Protect".

- Per creare snapshot dei volumi correlati di un'applicazione nello stesso momento, definisci i gruppi di volumi nell'applicazione. Fai riferimento a "Utilizza gli snapshot dei gruppi di volumi con Trident Protect".

Crea un'istantanea on-demand

Puoi creare uno snapshot on-demand in qualsiasi momento.

NOTA Le risorse soggette a ambito cluster sono incluse in un backup, in uno snapshot o in un clone, se fanno riferimento esplicitamente nella definizione dell'applicazione o se hanno riferimenti a uno qualsiasi dei namespace delle applicazioni.

Creare un'istantanea utilizzando una CR

Fasi

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-snapshot-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.applicationRef**: Il nome Kubernetes dell'applicazione da snapshot.
   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui devono essere memorizzati i contenuti (metadati) dello snapshot.
   - **Spec.reclaimPolicy**: (*Optional*) definisce cosa accade all'AppArchive di uno snapshot quando lo snapshot CR viene eliminato. Ciò significa che anche se impostato su `Retain`, l'istantanea verrà eliminata. Opzioni valide:
     - `Retain` (impostazione predefinita)
     - `Delete`

```
apiVersion: protect.trident.netapp.io/v1
kind: Snapshot
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  reclaimPolicy: Delete
```

3. Dopo aver popolato il `trident-protect-snapshot-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-snapshot-cr.yaml
```

Creare una snapshot utilizzando la CLI

Fasi

1. Creare l'istantanea, sostituendo i valori tra parentesi con le informazioni dell'ambiente. Ad esempio:

```
tridentctl-protect create snapshot <my_snapshot_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> -n
<application_namespace>
```

Crea un backup su richiesta

Puoi eseguire il backup di un'app in qualsiasi momento.

NOTA	Le risorse soggette a ambito cluster sono incluse in un backup, in uno snapshot o in un clone, se fanno riferimento esplicitamente nella definizione dell'applicazione o se hanno riferimenti a uno qualsiasi dei namespace delle applicazioni.

Prima di iniziare

Assicurati che la scadenza del token di sessione AWS sia sufficiente per eventuali operazioni di backup S3 a esecuzione prolungata. Se il token scade durante l'operazione di backup, l'operazione potrebbe non riuscire.

- Per ulteriori informazioni sulla verifica della scadenza corrente del token di sessione, fare riferimento "Documentazione di API AWS" al .
- Per ulteriori informazioni sulle credenziali con le risorse AWS, fare riferimento al "Documentazione di AWS IAM".

Creare un backup utilizzando una CR

Fasi

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-backup-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.applicationRef**: (*required*) il nome Kubernetes dell'applicazione di cui eseguire il backup.
   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui devono essere memorizzati i contenuti di backup.
   - **Spec.dataMover**: (*Optional*) stringa che indica quale strumento di backup utilizzare per l'operazione di backup. Valori possibili (distinzione tra maiuscole e minuscole):
     - `Restic`
     - `Kopia` (impostazione predefinita)
   - **Spec.reclaimPolicy**: (*Optional*) definisce cosa accade a un backup quando viene rilasciato dalla relativa dichiarazione. Valori possibili:
     - `Delete`

        - `Retain` (impostazione predefinita)
    - **spec.snapshotRef**: (*Facoltativo*): Nome dello snapshot da utilizzare come origine del backup. Se non viene fornito, verrà creato e eseguito il backup di uno snapshot temporaneo.

      Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: Kopia
```

3. Dopo aver popolato il `trident-protect-backup-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-backup-cr.yaml
```

Creare un backup utilizzando l'interfaccia CLI

Fasi

1. Creare il backup, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente. Ad esempio:

```
tridentctl-protect create backup <my_backup_name> --appvault <my-
vault-name> --app <name_of_app_to_back_up> --data-mover
<Kopia_or_Restic> -n <application_namespace>
```

   È possibile utilizzare il `--full-backup` flag per specificare se un backup deve essere non incrementale. Per impostazione predefinita, tutti i backup sono incrementali. Quando si utilizza questo indicatore, il backup diventa non incrementale. È consigliabile eseguire periodicamente un backup completo, quindi eseguire backup incrementali tra un backup completo e l'altro, in modo da ridurre al minimo il rischio associato ai ripristini.

Annotazioni di backup supportate

Nella tabella seguente vengono descritte le annotazioni che è possibile utilizzare durante la creazione di un CR di backup:

Annotazione	Tipo	Descrizione	Valore predefinito
protect.trident.netapp.io/full-backup	stringa	Specifica se un backup deve essere non incrementale. Impostato su `true` per creare un backup non incrementale. È consigliabile eseguire periodicamente un backup completo e poi eseguire backup incrementali tra un backup completo e l'altro, per ridurre al minimo i rischi associati ai ripristini.	"falso"
protect.trident.netapp.io/snapshot-completion-timeout	stringa	Tempo massimo consentito per il completamento dell'intera operazione di snapshot.	"60 metri"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	stringa	Tempo massimo consentito affinché gli snapshot del volume raggiungano lo stato pronto all'uso.	"30 metri"
protect.trident.netapp.io/volume-snapshots-created-timeout	stringa	Tempo massimo consentito per la creazione di snapshot del volume.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	stringa	Tempo massimo (in secondi) di attesa affinché i nuovi PersistentVolumeClaim (PVC) creati raggiungano il `Bound` fase prima del fallimento delle operazioni.	"1200" (20 minuti)

Creare un piano di data Protection

Una policy di protezione protegge un'app creando snapshot, backup o entrambi secondo una pianificazione definita. È possibile scegliere di creare snapshot e backup orari, giornalieri, settimanali e mensili e specificare il numero di copie da conservare. È possibile pianificare un backup completo non incrementale utilizzando l'annotazione full-backup-rule. Per impostazione predefinita, tutti i backup sono incrementali. L'esecuzione periodica di un backup completo, insieme a backup incrementali intermedi, aiuta a ridurre il rischio associato ai ripristini.

NOTA

- È possibile creare pianificazioni solo per gli snapshot impostando `backupRetention` a zero e `snapshotRetention` a un valore maggiore di zero. Collocamento `snapshotRetention` a zero significa che tutti i backup pianificati creeranno comunque degli snapshot, ma questi saranno temporanei e verranno eliminati immediatamente dopo il completamento del backup.
- Le risorse soggette a ambito cluster sono incluse in un backup, in uno snapshot o in un clone, se fanno riferimento esplicitamente nella definizione dell'applicazione o se hanno riferimenti a uno qualsiasi dei namespace delle applicazioni.

Creare una pianificazione utilizzando una CR

Fasi

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-schedule-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.

- **Spec.dataMover**: (*Optional*) stringa che indica quale strumento di backup utilizzare per l'operazione di backup. Valori possibili (distinzione tra maiuscole e minuscole):
  - `Restic`
  - `Kopia` (impostazione predefinita)
- **Spec.applicationRef**: Il nome Kubernetes dell'applicazione di cui eseguire il backup.
- **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui devono essere memorizzati i contenuti di backup.
- **spec.backupRetention**: (*Obbligatorio*) Numero di backup da conservare. Zero indica che non devono essere creati backup (solo snapshot).
- **spec.backupReclaimPolicy**: (*Facoltativo*) Determina cosa succede a un backup se il CR di backup viene eliminato durante il periodo di conservazione. Dopo il periodo di conservazione, i backup vengono sempre eliminati. Valori possibili (con distinzione tra maiuscole e minuscole):
  - `Retain` (impostazione predefinita)
  - `Delete`
- **spec.snapshotRetention**: (*Obbligatorio*) Numero di snapshot da conservare. Zero indica che non devono essere creati snapshot.
- **spec.snapshotReclaimPolicy**: (*Facoltativo*) Determina cosa succede a uno snapshot se il CR dello snapshot viene eliminato durante il periodo di conservazione. Dopo il periodo di conservazione, gli snapshot vengono sempre eliminati. Valori possibili (con distinzione tra maiuscole e minuscole):
  - `Retain`
  - `Delete`(predefinito)
- **spec.granularity**: frequenza di esecuzione della pianificazione. Valori possibili, insieme ai campi associati obbligatori:
  - `Hourly`(richiede che tu specifichi `spec.minute`)
  - `Daily`(richiede che tu specifichi `spec.minute` E `spec.hour`)
  - `Weekly`(richiede che tu specifichi `spec.minute, spec.hour` , E `spec.dayOfWeek`)
  - `Monthly`(richiede che tu specifichi `spec.minute, spec.hour` , E `spec.dayOfMonth`)
  - `Custom`
- **spec.dayOfMonth**: (*Facoltativo*) Il giorno del mese (1 - 31) in cui la pianificazione deve essere eseguita. Questo campo è obbligatorio se la granularità è impostata su `Monthly` . Il valore deve essere fornito come stringa.
- **spec.dayOfWeek**: (*Facoltativo*) Il giorno della settimana (0 - 7) in cui deve essere eseguita la pianificazione. I valori 0 o 7 indicano domenica. Questo campo è obbligatorio se la granularità è impostata su `Weekly` . Il valore deve essere fornito come stringa.
- **spec.hour**: (*Facoltativo*) L'ora del giorno (0 - 23) in cui la pianificazione deve essere eseguita. Questo campo è obbligatorio se la granularità è impostata su `Daily` , `Weekly` , O `Monthly` . Il valore deve essere fornito come stringa.
- **spec.minute**: (*Facoltativo*) Il minuto dell'ora (0 - 59) in cui la pianificazione deve essere eseguita. Questo campo è obbligatorio se la granularità è impostata su `Hourly` , `Daily` , `Weekly` , O `Monthly` . Il valore deve essere fornito come stringa.

- **spec.runImmediately**: (*Facoltativo*) Impostare su `true` per attivare un'esecuzione di base immediata e una tantum (backup e/o snapshot in base alle impostazioni di conservazione) quando viene creata la pianificazione. Il valore predefinito è `false`. Questo non modifica la ricorrenza successiva.

  Esempio di YAML per la pianificazione di backup e snapshot:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: Daily
  hour: "0"
  minute: "0"
```

  Esempio di YAML per la pianificazione solo snapshot:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-snapshot-schedule
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "0"
  snapshotRetention: "15"
  granularity: Daily
  hour: "2"
  minute: "0"
```

  Esempio YAML per una pianificazione con esecuzione immediata:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
```

```
metadata:
  namespace: my-app-namespace
  name: my-daily-schedule-run-immediately
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "7"
  snapshotRetention: "7"
  granularity: Daily
  hour: "3"
  minute: "0"
  runImmediately: true
```

3. Dopo aver popolato il `trident-protect-schedule-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

Creare una pianificazione utilizzando l'interfaccia CLI

Fasi

1. Creare il programma di protezione, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente. Ad esempio:

NOTA	È possibile utilizzare `tridentctl-protect create schedule --help` per visualizzare informazioni dettagliate sulla guida per questo comando.

```
tridentctl-protect create schedule <my_schedule_name> \
  --appvault <my_appvault_name> \
  --app <name_of_app_to_snapshot> \
  --backup-retention <how_many_backups_to_retain> \
  --backup-reclaim-policy <Retain|Delete (default Retain)> \
  --data-mover <Kopia_or_Restic> \
  --day-of-month <day_of_month_to_run_schedule> \
  --day-of-week <day_of_week_to_run_schedule> \
  --granularity <frequency_to_run> \
  --hour <hour_of_day_to_run> \
  --minute <minute_of_hour_to_run> \
  --recurrence-rule <recurrence> \
  --snapshot-retention <how_many_snapshots_to_retain> \
  --snapshot-reclaim-policy <Retain|Delete (default Delete)> \
  --full-backup-rule <string> \
  --run-immediately <true|false> \
  -n <application_namespace>
```

I seguenti flag forniscono un controllo aggiuntivo sulla tua pianificazione:

- **Pianificazione del backup completo**: utilizzare `--full-backup-rule` flag per pianificare backup completi non incrementali. Questa bandiera funziona solo con `--granularity Daily`. Valori possibili:
  - `Always`: Crea un backup completo ogni giorno.
  - Giorni feriali specifici: specificare uno o più giorni separati da virgole (ad esempio, `"Monday,Thursday"`). Valori validi: lunedì, martedì, mercoledì, giovedì, venerdì, sabato, domenica.

  NOTA IL `--full-backup-rule` il flag non funziona con la granularità oraria, settimanale o mensile.

- **Protezione di base immediata**: utilizza `--run-immediately true` per creare un backup o uno snapshot iniziale immediatamente quando viene creata la pianificazione, invece di attendere il primo orario di esecuzione pianificato. L'impostazione predefinita è `false`.
- **Programmazioni solo snapshot**: Imposta `--backup-retention 0` e specificare un valore maggiore di zero per `--snapshot-retention`.

Annotazioni di pianificazione supportate

Nella tabella seguente vengono descritte le annotazioni che è possibile utilizzare durante la creazione di una CR di pianificazione:

Annotazione	Tipo	Descrizione	Valore predefinito
protect.trident.netapp.io/full-backup-rule	stringa	Specifica la regola per la pianificazione dei backup completi. Puoi impostarlo su `Always` per un backup completo costante o personalizzarlo in base alle tue esigenze. Ad esempio, se si sceglie la granularità giornaliera, è possibile specificare i giorni feriali in cui deve essere eseguito il backup completo (ad esempio, `"Monday,Thursday"`). I valori validi per i giorni feriali sono: lunedì, martedì, mercoledì, giovedì, venerdì, sabato, domenica. Si noti che questa annotazione può essere utilizzata solo con pianificazioni che hanno `granularity` impostato su `Daily`.	Non impostato (tutti i backup sono incrementali)
protect.trident.netapp.io/snapshot-completion-timeout	stringa	Tempo massimo consentito per il completamento dell'intera operazione di snapshot.	"60 metri"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	stringa	Tempo massimo consentito affinché gli snapshot del volume raggiungano lo stato pronto all'uso.	"30 metri"
protect.trident.netapp.io/volume-snapshots-created-timeout	stringa	Tempo massimo consentito per la creazione di snapshot del volume.	"5m"

Annotazione	Tipo	Descrizione	Valore predefinito
protect.trident.netapp.io/pvc-bind-timeout-sec	stringa	Tempo massimo (in secondi) di attesa affinché i nuovi PersistentVolumeClaim (PVC) creati raggiungano il `Bound` fase prima del fallimento delle operazioni.	"1200" (20 minuti)

Eliminare uno snapshot

Eliminare le snapshot pianificate o on-demand non più necessarie.

Fasi

1. Rimuovere l'istantanea CR associata all'istantanea:

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

Eliminare un backup

Eliminare i backup pianificati o on-demand non più necessari.

NOTA Assicurati che la politica di recupero sia impostata su `Delete` per rimuovere tutti i dati di backup dall'archiviazione degli oggetti. L'impostazione predefinita del criterio è `Retain` per evitare la perdita accidentale di dati. Se la politica non viene modificata in `Delete` , i dati di backup rimarranno nell'archivio oggetti e richiederanno l'eliminazione manuale.

Fasi

1. Rimuovere il CR di backup associato al backup:

```
kubectl delete backup <backup_name> -n my-app-namespace
```

Controllare lo stato di un'operazione di backup

È possibile utilizzare la riga di comando per verificare lo stato di un'operazione di backup in corso, completata o non riuscita.

Fasi

1. Utilizzare il seguente comando per recuperare lo stato dell'operazione di backup, sostituendo i valori nei brackes con le informazioni dal proprio ambiente:

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath
='{.status}'
```

Abilitare backup e ripristino per operazioni Azure-NetApp-Files (ANF)

Se hai installato Trident Protect, puoi abilitare la funzionalità di backup e ripristino a basso consumo di spazio per i backend di archiviazione che utilizzano la classe di archiviazione azure-netapp-files e sono stati creati

prima di Trident 24.06. Questa funzionalità funziona con volumi NFSv4 e non consuma spazio aggiuntivo dal pool di capacità.

Prima di iniziare

Verificare quanto segue:

- Hai installato Trident Protect.
- Hai definito un'applicazione in Trident Protect. Questa applicazione avrà funzionalità di protezione limitate finché non avrai completato questa procedura.
- È stata `azure-netapp-files` selezionata come classe di archiviazione predefinita per il backend di archiviazione.

▼ **Espandere per la procedura di configurazione**

1. Se il volume ANF è stato creato prima dell'aggiornamento a Trident 24,10, procedere come segue in Trident:

   a. Abilitare la directory snapshot per ogni PV basata su file Azure-NetApp e associata all'applicazione:

   ```
   tridentctl update volume <pv name> --snapshot-dir=true -n trident
   ```

   b. Confermare che la directory snapshot è stata abilitata per ogni PV associato:

   ```
   tridentctl get volume <pv name> -n trident -o yaml | grep
   snapshotDir
   ```

   Risposta:

   ```
   snapshotDirectory: "true"
   ```

   +
   Quando la directory degli snapshot non è abilitata, Trident Protect sceglie la funzionalità di backup normale, che consuma temporaneamente spazio nel pool di capacità durante il processo di backup. In questo caso, assicurarsi che nel pool di capacità sia disponibile spazio sufficiente per creare un volume temporaneo delle dimensioni del volume sottoposto a backup.

Risultato

L'applicazione è pronta per il backup e il ripristino tramite Trident Protect. Ogni PVC può essere utilizzato anche da altre applicazioni per backup e ripristini.

Utilizza gli snapshot dei gruppi di volumi con Trident Protect

Utilizza gli snapshot dei gruppi di volumi per acquisire le richieste di volumi persistenti (PVC) correlate nello stesso momento. Questa funzionalità è utile per le applicazioni che scrivono su più volumi, incluse le macchine virtuali (VM) con più dischi.

A proposito di questa attività

Definisci i gruppi di volumi nel `spec.volumeGroups` campo della `Application` risorsa personalizzata (CR).

Quando crei uno snapshot dell'applicazione, Trident Protect utilizza un gruppo di coerenza ONTAP per creare uno snapshot di tutti i PVC in un gruppo di volumi nello stesso momento. Crea un gruppo di coerenza per ogni namespace nel gruppo di volumi. Ogni volume nel gruppo ottiene anche un proprio snapshot, così puoi ripristinare i volumi singolarmente.

Trident Protect crea istantanee individuali delle PVC che non appartengono a un gruppo di volumi. Queste istantanee vengono acquisite una dopo l'altra, quindi non è garantito che riflettano lo stesso preciso istante.

Gli snapshot di gruppo di volumi si applicano agli snapshot su richiesta, agli snapshot pianificati e allo snapshot che Trident Protect acquisisce come parte di un backup. Se esegui il backup di uno snapshot esistente, Trident Protect utilizza tale snapshot e non crea un nuovo snapshot di gruppo. Anche la replica non crea un altro snapshot di gruppo. Utilizza gli snapshot per volume già creati dallo snapshot di gruppo.

Trident Protect acquisisce i volumi nello stesso momento solo se soddisfano tutte le seguenti condizioni:

- Appartengono allo stesso gruppo di volumi.
- Si trovano nello stesso namespace.
- Si trovano sulla stessa macchina virtuale di storage (SVM).
- Il gruppo di volumi non contiene più di 80 volumi in quel namespace.

Se Trident Protect non riesce a creare uno snapshot di gruppo, ad esempio perché un requisito non è soddisfatto, aggiunge un avviso a `status.warnings` e crea snapshot individuali dei PVC di quel gruppo. Questi snapshot vengono acquisiti uno dopo l'altro, quindi potrebbero non riflettere lo stesso momento nel tempo. Lo snapshot dell'applicazione si completa comunque.

La tabella seguente descrive cosa accade nelle situazioni più comuni.

Situazione	Risultato
Il gruppo di volumi seleziona i PVC in più di un namespace	Un'istantanea di gruppo appartiene a un solo namespace, quindi Trident Protect crea un'istantanea di gruppo separata in ogni namespace. Queste istantanee di gruppo non vengono acquisite contemporaneamente. Una può avere successo mentre un'altra è ancora in esecuzione, fallisce o ricorre a istantanee individuali. Ad esempio, i PVC etichettati `app: payments-db` in `payments-a` e `payments-b` diventano due istantanee di gruppo separate. Non considerare le istantanee di gruppo provenienti da namespace diversi come un unico punto di ripristino.
I PVC in un namespace sono su più di una SVM	Trident Protect non suddivide i volumi per SVM o per Trident backend. Un gruppo di coerenza ONTAP non può includere volumi di più di una SVM, quindi lo snapshot di gruppo non riesce per l'intero namespace, inclusi i volumi che condividono una SVM. Trident Protect aggiunge un avviso, rimuove lo snapshot di gruppo e crea uno snapshot individuale di ciascun volume. Per acquisire i volumi insieme, mettili sulla stessa SVM.
Un namespace ha più di 80 PVC corrispondenti	Trident Protect ignora solo quel namespace e crea snapshot individuali dei relativi PVC. Gli altri namespace nello stesso gruppo di volumi ottengono comunque uno snapshot di gruppo.
Un PVC corrisponde a più di una `volumeGroups` voce	Trident Protect aggiunge il PVC a entrambi i gruppi di volumi e uno degli snapshot di gruppo può non riuscire. Lo snapshot di gruppo che non riesce ripiega su snapshot individuali. Assicurati che ogni PVC corrisponda a una sola voce.

Situazione	Risultato
Un gruppo di volumi include PVC che utilizzano un driver o un protocollo che non supporta gli snapshot dei gruppi di volumi	Lo snapshot di gruppo non riesce e Trident Protect crea snapshot individuali di tali PVC.

Prima di iniziare

- I tuoi PVC devono utilizzare un driver di storage e un protocollo che supportino gli snapshot dei gruppi di volumi. Per i driver e i protocolli supportati, consulta "Lavora con gli snapshot dei gruppi di volumi di Trident".
- Il cluster deve eseguire Kubernetes 1.36 o versioni successive.
- Il controller snapshot di Kubernetes deve essere alla versione 8.5.0 o successiva, con `CSIVolumeGroupSnapshot`feature gate abilitato.
- I `v1` CRD per `VolumeGroupSnapshot`, `VolumeGroupSnapshotClass` e `VolumeGroupSnapshotContent` devono essere installati.
- Devi usare Trident 26.10 o versione successiva con un backend online. Il `CSIVolumeGroupSnapshot` feature gate deve inoltre essere abilitato sul container Trident `csi-snapshotter`.
- È necessario definire una `VolumeGroupSnapshotClass` che utilizzi il driver `csi.trident.netapp.io`. Se viene definita una sola classe, Trident Protect la utilizza. Se vengono definite più classi, annota la classe che Trident Protect deve utilizzare con `snapshot.storage.kubernetes.io/is-default-class: "true"`. Altrimenti, Trident Protect utilizza la prima classe in ordine alfabetico. Non puoi scegliere la classe nel `Application` o `Snapshot` CR. Per ulteriori informazioni, consulta "Classe VolumeGroupSnapshot".
- L'account ONTAP nel backend Trident deve essere autorizzato a creare, ottenere ed eliminare gruppi di coerenza e a crearne snapshot. Fai riferimento a "Gestire i gruppi di coerenza delle applicazioni" e "Gestire gli snapshot dei gruppi di coerenza delle applicazioni". Il ruolo predefinito di amministratore SVM (`vsadmin`) non include queste autorizzazioni.

Definisci un gruppo di volumi

Trident Protect crea snapshot di gruppo solo per i volumi che appartengono a un gruppo di volumi. Aggiungi `spec.volumeGroups` al `Application` CR per definire i gruppi. Ogni voce rappresenta un gruppo separato. Seleziona i volumi per etichetta, per VM o per entrambi.

A proposito di questa attività

- È possibile definire gruppi di volumi solo nel `Application` CR. La `tridentctl-protect` CLI non dispone di un'opzione per i gruppi di volumi.
- Trident Protect include in un gruppo di volumi solo i PVC vincolati e crea snapshot individuali dei PVC non vincolati. Se una voce non seleziona alcun PVC vincolato, Trident Protect non crea uno snapshot di gruppo per tale voce.
- Assicurati che ogni PVC corrisponda a una sola voce.

Prima di iniziare

Definisci l'applicazione che desideri proteggere. Fai riferimento a "Definisci un'applicazione per la gestione con Trident Protect".

Fasi

1. Se selezioni i volumi tramite etichetta, etichetta i PVC che desideri raggruppare. Puoi utilizzare qualsiasi

chiave e valore di etichetta. Gli esempi in questa pagina utilizzano `volume-group=my-volume-group`.

```
kubectl label pvc <pvc_name_1> <pvc_name_2> volume-group=my-volume-group
-n <application_namespace>
```

2. Aggiungi `spec.volumeGroups` al tuo file CR `Application` esistente. Ogni voce deve includere `labelSelector`, `vmSelector` o entrambi. Se una voce include entrambi i campi, il gruppo di volumi include ogni volume selezionato da uno dei due campi.

   - **spec.volumeGroups[].labelSelector**: Un selettore di etichette Kubernetes (`matchLabels` o `matchExpressions`) che seleziona i PVC nell'applicazione. Un selettore vuoto non è valido.
   - **spec.volumeGroups[].vmSelector**: Un elenco di VM KubeVirt, ciascuna con un `namespace` e `name`. Trident Protect include i PVC che ogni VM monta.

     L'esempio seguente definisce due gruppi di volumi. Il primo gruppo comprende i PVC etichettati `volume-group: my-volume-group`. Il secondo gruppo comprende i PVC che `my-vm-1` e `my-vm-2` montano.

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-app-name
  namespace: my-namespace
spec:
  includedNamespaces:
    - namespace: my-namespace
  volumeGroups:
    - labelSelector:
        matchLabels:
          volume-group: my-volume-group
    - vmSelector:
        - namespace: my-namespace
          name: my-vm-1
        - namespace: my-namespace
          name: my-vm-2
```

+

NOTA Mantieni gli altri campi che il tuo `Application` CR contiene già, come `spec.resourceFilter`. Se ometti un campo dal file che applichi, `kubectl apply` lo elimina dal CR.

3. Applica il CR aggiornato:

```
kubectl apply -f <application_cr_file>.yaml
```

Risultato

Il `Application` CR include i gruppi di volumi. La successiva istantanea dell'applicazione li utilizza.

Creare un'istantanea

Crea uno snapshot su richiesta, un backup su richiesta o una pianificazione della protezione dei dati per l'applicazione. Non è necessario aggiungere alcun campo. Trident Protect utilizza i gruppi di volumi che hai definito nel `Application` CR. Per istruzioni, consulta "Proteggi le applicazioni utilizzando Trident Protect".

NOTA Per impostazione predefinita, Trident Protect crea lo snapshot di gruppo prima di congelare i filesystem delle VM. I volumi in un gruppo di volumi vengono acquisiti senza congelamento, quindi i dati che una VM non ha ancora scritto su disco non sono inclusi. I dischi che non sono nel gruppo di volumi vengono acquisiti in seguito, quando Trident Protect crea snapshot individuali dopo aver congelato la VM. Per acquisire tutti i dischi di una VM nello stesso momento, includi tutti i dischi nel gruppo di volumi. Per ulteriori informazioni sul congelamento dei filesystem delle VM, consulta "Protezione dei dati con le macchine virtuali KubeVirt".

Rivedi il risultato

Al termine dell'acquisizione dello snapshot, controlla il relativo stato per confermare che Trident Protect abbia creato gli snapshot di gruppo.

NOTA Il completamento di uno snapshot di un'applicazione non significa che gli snapshot di gruppo siano andati a buon fine. La `VolumeGroupSnapshotsCreated` condizione diventa `true` anche se ogni snapshot di gruppo viene saltato o non riesce, e Trident Protect non crea un evento o una metrica di Kubernetes. Per rilevarlo, monitora `status.warnings` e i `skipped` campi e `failed`.

Fasi

1. Ottieni il nome dello snapshot. Per elencare gli snapshot dell'applicazione, esegui il seguente comando:

```
kubectl get snapshots -n <application_namespace>
```

2. Verifica lo stato dell'istantanea:

```
kubectl get snapshots <snapshot_name> -n <application_namespace> -o yaml
```

Risultato

Il `status.volumeGroupSnapshots` campo contiene una voce per ogni snapshot di gruppo. Trident Protect crea uno snapshot di gruppo separato per ogni namespace in un gruppo di volumi. L'esempio seguente è abbreviato:

```
status:
  volumeGroupSnapshots:
```

```
    - name: vgs-<snapshot_uid_prefix>-<group_index>-<namespace>
      namespace: <namespace>
      ready: true
      memberPVCs:
        - pvcName: <pvc_name_1>
          pvcNamespace: <namespace>
        - pvcName: <pvc_name_2>
          pvcNamespace: <namespace>
```

Il nome dello snapshot di gruppo include i primi otto caratteri dell' `Snapshot`UID, l'indice della `spec.volumeGroups` voce (a partire da `0`), e il namespace, troncato a 12 caratteri.

- `ready: true` significa che lo snapshot del gruppo è stato creato correttamente. Gli snapshot che compongono il gruppo sono elencati anche sotto `status.volumeSnapshots`. Si tratta degli snapshot membri del gruppo, non di snapshot aggiuntivi degli stessi volumi.
- `skipped: true` significa che Trident Protect non ha tentato di creare lo snapshot del gruppo, perché il gruppo di volumi ha più di 80 PVC in quel namespace. Trident Protect crea invece snapshot individuali dei volumi del gruppo e aggiunge il `skipReason` testo a `status.warnings`.
- `failed: true` significa che Trident Protect ha tentato di creare lo snapshot di gruppo, ma l'operazione non è riuscita. Trident Protect crea invece snapshot individuali dei volumi del gruppo e aggiunge il `error` testo a `status.warnings`.

Etichette PVC temporanee e pulizia

Trident Protect aggiunge un'etichetta a ogni PVC in uno snapshot di gruppo, ad esempio `protect.trident.netapp.io/vgs-group-673c0189-0=true`. Le tue etichette non vengono modificate. L'etichetta rimane sul PVC finché non elimini lo snapshot dell'applicazione. Quando elimini lo snapshot dell'applicazione, Trident Protect elimina anche i relativi snapshot di gruppo di volumi e gli snapshot dei membri e rimuove le etichette dai PVC. Questo avviene indipendentemente dal `reclaimPolicy` dello snapshot sia `Delete` o `Retain`.

Quali sono le prossime novità?

Ripristina l'applicazione dallo snapshot nello stesso modo in cui faresti per qualsiasi altro snapshot. Gli snapshot dei gruppi di volumi non hanno un ripristino separato. Trident Protect esegue il ripristino dagli snapshot membri, quindi puoi ripristinare singoli PVC o VM senza ripristinare ogni PVC del gruppo. Fai riferimento a "Ripristino da uno snapshot a uno spazio dei nomi diverso" o "Ripristinare da uno snapshot allo spazio dei nomi originale".

Informazioni correlate

- "Definisci un'applicazione per la gestione con Trident Protect"
- "Proteggi le applicazioni utilizzando Trident Protect"
- "Proteggi le macchine virtuali KubeVirt con Trident Protect"

Ripristino delle applicazioni

Ripristina le applicazioni utilizzando Trident Protect

Puoi utilizzare Trident Protect per ripristinare la tua applicazione da uno snapshot o da un backup. Il ripristino da uno snapshot esistente sarà più rapido se si ripristina

l'applicazione nello stesso cluster.

NOTA

- Quando si ripristina un'applicazione, tutti i collegamenti di esecuzione configurati per l'applicazione vengono ripristinati con l'applicazione. Se è presente un gancio di esecuzione post-ripristino, viene eseguito automaticamente come parte dell'operazione di ripristino.
- Il ripristino da un backup a un namespace diverso o al namespace originale è supportato per i volumi qtree. Tuttavia, il ripristino da uno snapshot a un namespace diverso o al namespace originale non è supportato per i volumi qtree.
- È possibile utilizzare le impostazioni avanzate per personalizzare le operazioni di ripristino. Per saperne di più, fare riferimento a "Utilizza le impostazioni di ripristino avanzate Trident Protect".
- Per ripristinare specifiche macchine virtuali (VM) KubeVirt da un archivio senza ripristinare tutte le risorse dello spazio dei nomi, usa il campo `includedVirtualMachines` su una CR di ripristino. Per le CR supportate, il riferimento ai campi e gli esempi YAML, consulta "Proteggi le macchine virtuali KubeVirt con Trident Protect".

Ripristino da un backup a uno spazio dei nomi diverso

Quando si ripristina un backup in uno spazio dei nomi diverso utilizzando un CR BackupRestore, Trident Protect ripristina l'applicazione in un nuovo spazio dei nomi e crea un CR dell'applicazione per l'applicazione ripristinata. Per proteggere l'applicazione ripristinata, creare backup o snapshot su richiesta oppure stabilire una pianificazione di protezione.

NOTA

- Il ripristino di un backup in uno spazio dei nomi diverso con le risorse esistenti non altererà le risorse che condividono i nomi con quelli del backup. Per ripristinare tutte le risorse del backup, eliminare e ricreare lo spazio dei nomi di destinazione o ripristinare il backup in un nuovo spazio dei nomi.
- Quando si utilizza una CR per ripristinare un nuovo namespace, è necessario creare manualmente il namespace di destinazione prima di applicare la CR. Trident Protect crea automaticamente gli spazi dei nomi solo quando si utilizza la CLI.

Prima di iniziare

Assicurati che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino S3 con esecuzione prolungata. Se il token scade durante l'operazione di ripristino, l'operazione potrebbe non riuscire.

- Per ulteriori informazioni sulla verifica della scadenza corrente del token di sessione, fare riferimento "Documentazione di API AWS" al .
- Per ulteriori informazioni sulle credenziali con le risorse AWS, fare riferimento al "Documentazione di AWS IAM".

NOTA

Quando si ripristinano i backup utilizzando Kopia come strumento di spostamento dati, è possibile specificare facoltativamente annotazioni nel CR o tramite la CLI per controllare il comportamento dell'archiviazione temporanea utilizzata da Kopia. Fare riferimento al "Documentazione Kopia" per maggiori informazioni sulle opzioni che puoi configurare. Utilizzare il `tridentctl-protect create --help` comando per ulteriori informazioni sulla specifica delle annotazioni con la CLI Trident Protect.

Utilizzare un CR

Fasi

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-backup-restore-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono memorizzati i contenuti di backup. Per trovare il percorso, utilizzare il seguente comando:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui sono memorizzati i contenuti di backup.
   - **spec.destinationApplicationName**: (*Facoltativo*) Il nome per l'applicazione ripristinata. Se specificato, l'applicazione ripristinata utilizza questo nome. Se non specificato, l'applicazione ripristinata utilizza il nome dell'applicazione di origine.
   - **spec.namespaceMapping**: mappatura dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituire `my-source-namespace` e `my-destination-namespace` con le informazioni del proprio ambiente.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: BackupRestore
     metadata:
       name: my-cr-name
       namespace: my-destination-namespace
     spec:
       appArchivePath: my-backup-path
       appVaultRef: appvault-name
       destinationApplicationName: my-new-app-name
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. (*Optional*) se è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda risorse contrassegnate con determinate etichette:

   NOTA Trident Protect seleziona automaticamente alcune risorse in base alla loro relazione con le risorse selezionate. Ad esempio, se selezioni una risorsa di richiesta di volume persistente e a questa è associato un pod, Trident Protect ripristinerà anche il pod associato.

   - **ResourceFilter.resourceSelectionCriteria**: (Necessario per il filtraggio) utilizzare `Include` o includere o `Exclude` escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti

parametri resourceMatcher per definire le risorse da includere o escludere:

- **ResourceFilter.resourceMatchers**: Una matrice di oggetti resourceMatcher. Se si definiscono più elementi in questa matrice, questi corrispondono come un'operazione OR e i campi all'interno di ogni elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.
  - **ResourceMatchers[].group**: (*Optional*) Gruppo della risorsa da filtrare.
  - **ResourceMatchers[].Kind**: (*Optional*) tipo di risorsa da filtrare.
  - **ResourceMatchers[].version**: (*Optional*) versione della risorsa da filtrare.
  - **ResourceMatchers[].names**: (*Optional*) nomi nel campo Kubernetes metadata.name della risorsa da filtrare.
  - **ResourceMatchers[].namespaces**: (*Optional*) Namespaces nel campo Kubernetes metadata.name della risorsa da filtrare.
  - **ResourceMatchers[].labelSelectors**: (*Optional*) stringa del selettore di etichette nel campo Kubernetes metadata.name della risorsa come definito nella "Documentazione Kubernetes" . Ad esempio: `"trident.netapp.io/os=linux"`.

    Ad esempio:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Dopo aver popolato il `trident-protect-backup-restore-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Utilizzare la CLI

Fasi

1. Ripristinare il backup su uno spazio dei nomi diverso, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente. L' `namespace-mapping`argomento utilizza spazi

dei nomi separati da due punti per mappare gli spazi dei nomi di origine agli spazi dei nomi di destinazione corretti nel formato `source1:dest1,source2:dest2`. Ad esempio:

```
tridentctl-protect create backuprestore <my_restore_name> \
--backup <backup_namespace>/<backup_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--destination-app-name<custom_app_name>\
-n <application_namespace>
```

Eseguire il ripristino da un backup nello spazio dei nomi originale

È possibile ripristinare un backup nello spazio dei nomi originale in qualsiasi momento. Quando si esegue un rispristino in-place, Trident Protect gestisce automaticamente le pianificazioni di protezione e le operazioni in corso per impedire punti di ripristino non validi:

- Tutte le pianificazioni di protezione abilitate per l'applicazione vengono disabilitate prima dell'inizio del ripristino. Ciò impedisce che backup o snapshot pianificati vengano eseguiti mentre le risorse dell'applicazione vengono ripristinate.
- Una volta completato correttamente il ripristino, solo le pianificazioni che erano abilitate prima del ripristino vengono riabilitate. Le pianificazioni che erano già disabilitate rimangono disabilitate.
- Tutte le operazioni di backup o snapshot in corso vengono annullate prima dell'inizio del ripristino. Se un'operazione non viene annullata entro 5 minuti, il ripristino procede e registra un avviso nello stato CR del ripristino.

Prima di iniziare

Assicurati che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino S3 con esecuzione prolungata. Se il token scade durante l'operazione di ripristino, l'operazione potrebbe non riuscire.

- Per ulteriori informazioni sulla verifica della scadenza corrente del token di sessione, fare riferimento "Documentazione di API AWS" al .
- Per ulteriori informazioni sulle credenziali con le risorse AWS, fare riferimento al "Documentazione di AWS IAM".

NOTA Quando si ripristinano i backup utilizzando Kopia come strumento di spostamento dati, è possibile specificare facoltativamente annotazioni nel CR o tramite la CLI per controllare il comportamento dell'archiviazione temporanea utilizzata da Kopia. Fare riferimento al "Documentazione Kopia" per maggiori informazioni sulle opzioni che puoi configurare. Utilizzare il `tridentctl-protect create --help` comando per ulteriori informazioni sulla specifica delle annotazioni con la CLI Trident Protect.

Utilizzare un CR

Fasi

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-backup-ipr-cr.yaml`.

2. Nel file creato, configurare i seguenti attributi:

   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono memorizzati i contenuti di backup. Per trovare il percorso, utilizzare il seguente comando:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui sono memorizzati i contenuti di backup.

     Ad esempio:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appArchivePath: my-backup-path
  appVaultRef: appvault-name
```

3. (*Optional*) se è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda risorse contrassegnate con determinate etichette:

NOTA	Trident Protect seleziona automaticamente alcune risorse in base alla loro relazione con le risorse selezionate. Ad esempio, se selezioni una risorsa di richiesta di volume persistente e a questa è associato un pod, Trident Protect ripristinerà anche il pod associato.

   - **ResourceFilter.resourceSelectionCriteria**: (Necessario per il filtraggio) utilizzare `Include` o includere o `Exclude` escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti parametri resourceMatcher per definire le risorse da includere o escludere:
     - **ResourceFilter.resourceMatchers**: Una matrice di oggetti resourceMatcher. Se si definiscono più elementi in questa matrice, questi corrispondono come un'operazione OR e i campi all'interno di ogni elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.
       - **ResourceMatchers[].group**: (*Optional*) Gruppo della risorsa da filtrare.
       - **ResourceMatchers[].Kind**: (*Optional*) tipo di risorsa da filtrare.
       - **ResourceMatchers[].version**: (*Optional*) versione della risorsa da filtrare.
       - **ResourceMatchers[].names**: (*Optional*) nomi nel campo Kubernetes metadata.name della risorsa da filtrare.
       - **ResourceMatchers[].namespaces**: (*Optional*) Namespaces nel campo Kubernetes metadata.name della risorsa da filtrare.

- **ResourceMatchers[].labelSelectors**: (*Optional*) stringa del selettore di etichette nel campo Kubernetes metadata.name della risorsa come definito nella "Documentazione Kubernetes" . Ad esempio: `"trident.netapp.io/os=linux"`.

  Ad esempio:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Dopo aver popolato il `trident-protect-backup-ipr-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

Utilizzare la CLI

Fasi

1. Ripristinare il backup nello spazio dei nomi originale, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente. L'`backup`argomento utilizza uno spazio dei nomi e un nome di backup nel formato `<namespace>/<name>`. Ad esempio:

```
tridentctl-protect create backupinplacerestore <my_restore_name> \
--backup <namespace/backup_to_restore> \
-n <application_namespace>
```

Ripristino da un backup a un cluster diverso

In caso di problemi con il cluster originale, è possibile ripristinare un backup su un cluster diverso.

NOTA

- Quando si ripristinano i backup utilizzando Kopia come strumento di spostamento dati, è possibile specificare facoltativamente annotazioni nel CR o tramite la CLI per controllare il

comportamento dell'archiviazione temporanea utilizzata da Kopia. Fare riferimento al "Documentazione Kopia" per maggiori informazioni sulle opzioni che puoi configurare. Utilizzare il `tridentctl-protect create --help` comando per ulteriori informazioni sulla specifica delle annotazioni con la CLI Trident Protect.

- Quando si utilizza una CR per ripristinare un nuovo namespace, è necessario creare manualmente il namespace di destinazione prima di applicare la CR. Trident Protect crea automaticamente gli spazi dei nomi solo quando si utilizza la CLI.

Prima di iniziare

Assicurarsi che siano soddisfatti i seguenti prerequisiti:

- Nel cluster di destinazione è installato Trident Protect.
- Il cluster di destinazione ha accesso al percorso bucket dello stesso AppVault del cluster di origine, dove è memorizzato il backup.
- Assicurarsi che l'ambiente locale possa connettersi al bucket di archiviazione degli oggetti definito in AppVault CR durante l'esecuzione di `tridentctl-protect get appvaultcontent` comando. Se le restrizioni di rete impediscono l'accesso, eseguire invece la CLI Trident Protect da un pod sul cluster di destinazione.
- Assicurati che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino con esecuzione prolungata. Se il token scade durante l'operazione di ripristino, l'operazione potrebbe non riuscire.
  - Per ulteriori informazioni sulla verifica della scadenza corrente del token di sessione, fare riferimento "Documentazione di API AWS" al .
  - Per ulteriori informazioni sulle credenziali con le risorse AWS, fare riferimento al "Documentazione AWS".

Fasi

1. Verifica che la AppVault CR esista sul cluster di destinazione utilizzando il plugin CLI di Trident Protect:

```
tridentctl-protect get appvault --context <destination_cluster_name>
```

NOTA Se il AppVault CR non esiste sul cluster di destinazione, crealo seguendo i passaggi in "Utilizzare gli oggetti Trident Protect AppVault per gestire i bucket".

2. Visualizza il contenuto del backup disponibile AppVault sul cluster di destinazione e prendi nota `appArchivePath` del backup che desideri ripristinare:

```
tridentctl-protect get appvaultcontent <appvault_name> \
--show-resources backup \
--show-paths \
--context <destination_cluster_name>
```

L'esecuzione di questo comando visualizza i backup disponibili in AppVault, inclusi i relativi cluster di origine, i nomi delle applicazioni corrispondenti, i timestamp e i percorsi di archivio.

Esempio di output:

```
+------------+----------+--------+----------------
+-------------------------+-------------+
|   CLUSTER   |    APP    |  TYPE  |      NAME       |        TIMESTAMP
|    PATH     |
+------------+----------+--------+----------------
+-------------------------+-------------+
| production1 | wordpress | backup | wordpress-bkup-1| 2024-10-30
08:37:40 (UTC)| backuppath1 |
| production1 | wordpress | backup | wordpress-bkup-2| 2024-10-30
08:37:40 (UTC)| backuppath2 |
+------------+----------+--------+----------------
+-------------------------+-------------+
```

3. Ripristinare l'applicazione nel cluster di destinazione utilizzando il nome AppVault e il percorso di archiviazione:

NOTA Quando si utilizza una CR, assicurarsi che lo spazio dei nomi destinato al ripristino dell'applicazione esista sul cluster di destinazione.

Utilizzare un CR

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-backup-restore-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui sono memorizzati i contenuti di backup.
   - **spec.appArchivePath**: (*Obbligatorio*) Il percorso all'interno di AppVault in cui sono archiviati i contenuti del backup. Utilizzare il comando del passaggio 2 per visualizzare i contenuti del backup e trovare `appArchivePath` per il backup che si desidera ripristinare.
   - **spec.destinationApplicationName**: (*Facoltativo*) Il nome per l'applicazione ripristinata. Se specificato, l'applicazione ripristinata utilizza questo nome. Se non specificato, l'applicazione ripristinata utilizza il nome dell'applicazione di origine.
   - **spec.namespaceMapping**: mappatura dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituire `my-source-namespace` e `my-destination-namespace` con le informazioni del proprio ambiente.

     Ad esempio:

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: my-cr-name
  namespace: my-destination-namespace
```

```
spec:
  appVaultRef: appvault-name
  appArchivePath: my-backup-path
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. Dopo aver popolato il `trident-protect-backup-restore-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Utilizzare la CLI

1. Utilizzare il seguente comando per ripristinare l'applicazione, sostituendo i valori tra parentesi con le informazioni dell'ambiente. L'argomento namespace-mapping utilizza spazi dei nomi separati da due punti per mappare gli spazi dei nomi di origine agli spazi dei nomi di destinazione corretti nel formato source1:dest1,source2:dest2. Ad esempio:

```
tridentctl-protect create backuprestore <restore_name> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--appvault <appvault_name> \
--path <backup_path> \
--destination-app-name <custom_app_name> \
--context <destination_cluster_name> \
-n <application_namespace>
```

Ripristino da uno snapshot a uno spazio dei nomi diverso

È possibile ripristinare i dati da uno snapshot utilizzando un file di risorse personalizzato (CR) in uno spazio dei nomi diverso o nello spazio dei nomi di origine originale. Quando si ripristina uno snapshot in un namespace diverso utilizzando un CR SnapshotRestore, Trident Protect ripristina l'applicazione in un nuovo namespace e crea un CR per l'applicazione ripristinata. Per proteggere l'applicazione ripristinata, creare backup o snapshot su richiesta oppure stabilire una pianificazione di protezione.

NOTA

- SnapshotRestore supporta il `spec.storageClassMapping` attributo, ma solo quando le classi di archiviazione di origine e di destinazione utilizzano lo stesso backend di archiviazione. Se si tenta di ripristinare un `StorageClass` che utilizza un backend di archiviazione diverso, l'operazione di ripristino non riuscirà.
- Quando si utilizza una CR per ripristinare un nuovo namespace, è necessario creare manualmente il namespace di destinazione prima di applicare la CR. Trident Protect crea automaticamente gli spazi dei nomi solo quando si utilizza la CLI.

Prima di iniziare

Assicurati che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino S3 con esecuzione prolungata. Se il token scade durante l'operazione di ripristino, l'operazione potrebbe non

riuscire.

- Per ulteriori informazioni sulla verifica della scadenza corrente del token di sessione, fare riferimento "Documentazione di API AWS" al .
- Per ulteriori informazioni sulle credenziali con le risorse AWS, fare riferimento al "Documentazione di AWS IAM".

Utilizzare un CR

Fasi

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-snapshot-restore-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui sono memorizzati i contenuti dello snapshot.
   - **Spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono memorizzati i contenuti dello snapshot. Per trovare il percorso, utilizzare il seguente comando:

     ```
     kubectl get snapshots <SNAPHOT_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.destinationApplicationName**: (*Facoltativo*) Il nome per l'applicazione ripristinata. Se specificato, l'applicazione ripristinata utilizza questo nome. Se non specificato, l'applicazione ripristinata utilizza il nome dell'applicazione di origine.
   - **spec.namespaceMapping**: mappatura dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituire `my-source-namespace` e `my-destination-namespace` con le informazioni del proprio ambiente.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: my-destination-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. (*Optional*) se è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda risorse contrassegnate con determinate etichette:

NOTA	Trident Protect seleziona automaticamente alcune risorse in base alla loro relazione con le risorse selezionate. Ad esempio, se selezioni una risorsa di richiesta di volume persistente e a questa è associato un pod, Trident Protect ripristinerà anche il pod associato.

- **ResourceFilter.resourceSelectionCriteria**: (Necessario per il filtraggio) utilizzare `Include` o includere o `Exclude` escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti parametri resourceMatcher per definire le risorse da includere o escludere:
  - **ResourceFilter.resourceMatchers**: Una matrice di oggetti resourceMatcher. Se si definiscono più elementi in questa matrice, questi corrispondono come un'operazione OR e i campi all'interno di ogni elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.
    - **ResourceMatchers[].group**: (*Optional*) Gruppo della risorsa da filtrare.
    - **ResourceMatchers[].Kind**: (*Optional*) tipo di risorsa da filtrare.
    - **ResourceMatchers[].version**: (*Optional*) versione della risorsa da filtrare.
    - **ResourceMatchers[].names**: (*Optional*) nomi nel campo Kubernetes metadata.name della risorsa da filtrare.
    - **ResourceMatchers[].namespaces**: (*Optional*) Namespaces nel campo Kubernetes metadata.name della risorsa da filtrare.
    - **ResourceMatchers[].labelSelectors**: (*Optional*) stringa del selettore di etichette nel campo Kubernetes metadata.name della risorsa come definito nella "Documentazione Kubernetes" . Ad esempio: `"trident.netapp.io/os=linux"`.

      Ad esempio:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Dopo aver popolato il `trident-protect-snapshot-restore-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Utilizzare la CLI

Fasi

1. Ripristinare lo snapshot in uno spazio dei nomi diverso, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente.

   - L'`snapshot`argomento utilizza uno spazio dei nomi e un nome snapshot nel formato `<namespace>/<name>.
   - L'`namespace-mapping`argomento utilizza spazi dei nomi separati da due punti per mappare gli spazi dei nomi di origine agli spazi dei nomi di destinazione corretti nel formato `source1:dest1,source2:dest2.

     Ad esempio:

```
tridentctl-protect create snapshotrestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--destination-app-name <custom_app_name> \
-n <application_namespace>
```

Ripristinare da uno snapshot allo spazio dei nomi originale

È possibile ripristinare uno snapshot nello spazio dei nomi originale in qualsiasi momento. Quando si esegue un rispristino in-place, Trident Protect gestisce automaticamente le pianificazioni di protezione e le operazioni in corso per impedire punti di ripristino non validi:

- Tutte le pianificazioni di protezione abilitate per l'applicazione vengono disabilitate prima dell'inizio del ripristino. Ciò impedisce che backup o snapshot pianificati vengano eseguiti mentre le risorse dell'applicazione vengono ripristinate.
- Una volta completato correttamente il ripristino, solo le pianificazioni che erano abilitate prima del ripristino vengono riabilitate. Le pianificazioni che erano già disabilitate rimangono disabilitate.
- Tutte le operazioni di backup o snapshot in corso vengono annullate prima dell'inizio del ripristino. Se un'operazione non viene annullata entro 5 minuti, il ripristino procede e registra un avviso nello stato CR del ripristino.

Prima di iniziare

Assicurati che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino S3 con esecuzione prolungata. Se il token scade durante l'operazione di ripristino, l'operazione potrebbe non riuscire.

- Per ulteriori informazioni sulla verifica della scadenza corrente del token di sessione, fare riferimento "Documentazione di API AWS" al .
- Per ulteriori informazioni sulle credenziali con le risorse AWS, fare riferimento al "Documentazione di AWS IAM".

Utilizzare un CR

Fasi

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-snapshot-ipr-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui sono memorizzati i contenuti dello snapshot.
   - **Spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono memorizzati i contenuti dello snapshot. Per trovare il percorso, utilizzare il seguente comando:

     ```
     kubectl get snapshots <SNAPSHOT_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

```
---
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appVaultRef: appvault-name
  appArchivePath: my-snapshot-path
```

+ . (*Facoltativo*) Se hai bisogno di selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungi un filtro che includa o escluda le risorse contrassegnate con etichette specifiche:

+ NOTA: Trident Protect seleziona automaticamente alcune risorse in base alla loro relazione con le risorse che selezioni. Ad esempio, se selezioni una risorsa di tipo persistent volume claim e questa ha un pod associato, Trident Protect ripristinerà anche il pod associato.

+ * **resourceFilter.resourceSelectionCriteria**: (Obbligatorio per il filtraggio) Usa `Include` o `Exclude` per includere o escludere una risorsa definita in resourceMatchers. Aggiungi i seguenti parametri resourceMatchers per definire le risorse da includere o escludere: **resourceFilter.resourceMatchers: Un array di oggetti resourceMatcher. Se definisci più elementi in questo array, corrispondono come un'operazione OR e i campi all'interno di ciascun elemento (group, kind, version) corrispondono come un'operazione AND.** * **resourceMatchers[].group**: (*Opzionale*) Gruppo della risorsa da filtrare. *** resourceMatchers[].kind: (*Opzionale*) Tipo della risorsa da filtrare.** * **resourceMatchers[].version**: (*Opzionale*) Versione della risorsa da filtrare. *** resourceMatchers[].names: (*Opzionale*) Nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.** * **resourceMatchers[].namespaces**: (*Opzionale*) Namespace nel campo metadata.name di Kubernetes della risorsa da filtrare. *** resourceMatchers[].labelSelectors**: (*Opzionale*) Stringa del selettore di etichette nel campo metadata.name di Kubernetes della risorsa come definito in "Documentazione Kubernetes". Ad esempio: `"trident.netapp.io/os=linux"`.

+ Ad esempio:

+

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

+ . Dopo aver inserito i valori corretti nel file `trident-protect-snapshot-ipr-cr.yaml`, applica il CR:

+

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

Utilizzare la CLI

Fasi

1. Ripristinare lo snapshot nello spazio dei nomi originale, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente. Ad esempio:

```
tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
-n <application_namespace>
```

Controllare lo stato di un'operazione di ripristino

È possibile utilizzare la riga di comando per verificare lo stato di un'operazione di ripristino in corso, completata o non riuscita.

Fasi

1. Utilizzare il seguente comando per recuperare lo stato dell'operazione di ripristino, sostituendo i valori nei brackes con le informazioni dall'ambiente in uso:

```
kubectl get backuprestore -n <namespace_name> <my_restore_cr_name> -o
jsonpath='{.status}'
```

Utilizza le impostazioni di ripristino avanzate Trident Protect

È possibile personalizzare le operazioni di ripristino utilizzando impostazioni avanzate quali annotazioni, impostazioni dello spazio dei nomi e opzioni di archiviazione per soddisfare esigenze specifiche.

Annotazioni ed etichette del namespace durante le operazioni di ripristino e failover

Durante le operazioni di ripristino e failover, vengono applicate etichette e annotazioni nel namespace di destinazione in modo che corrispondano alle etichette e alle annotazioni nel namespace di origine. Vengono aggiunte etichette o annotazioni dallo spazio dei nomi di origine che non esistono nello spazio dei nomi di destinazione e le etichette o annotazioni già esistenti vengono sovrascritte per corrispondere al valore dello spazio dei nomi di origine. Le etichette o le annotazioni presenti solo nello spazio dei nomi di destinazione rimangono invariate.

NOTA Se si utilizza Red Hat OpenShift, è importante tenere presente il ruolo fondamentale delle annotazioni dello spazio dei nomi negli ambienti OpenShift. Le annotazioni dello spazio dei nomi garantiscono che i pod ripristinati aderiscano alle autorizzazioni appropriate e alle configurazioni di sicurezza definite dai vincoli del contesto di sicurezza (SCC) di OpenShift e possano accedere ai volumi senza problemi di autorizzazione. Per maggiori informazioni, fare riferimento al"Documentazione dei vincoli del contesto di protezione OpenShift" .

Puoi impedire la sovrascrittura delle annotazioni specifiche nel namespace di destinazione impostando la variabile dell'ambiente Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` prima di eseguire l'operazione di ripristino o failover. Ad esempio:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

NOTA Quando si esegue un'operazione di ripristino o failover, tutte le annotazioni e le etichette dello spazio dei nomi specificate in `restoreSkipNamespaceAnnotations` E `restoreSkipNamespaceLabels` sono esclusi dall'operazione di ripristino o failover. Assicurarsi che queste impostazioni siano configurate durante l'installazione iniziale di Helm. Per saperne di più, fare riferimento a "Configurare le impostazioni aggiuntive del grafico del timone Trident Protect".

Se hai installato l'applicazione sorgente utilizzando Helm con `--create-namespace` bandiera, un trattamento speciale è riservato al `name` etichetta chiave. Durante il processo di ripristino o failover, Trident Protect copia questa etichetta nello spazio dei nomi di destinazione, ma aggiorna il valore al valore dello spazio dei nomi di destinazione se il valore dell'origine corrisponde allo spazio dei nomi di origine. Se questo valore non corrisponde allo spazio dei nomi di origine, viene copiato nello spazio dei nomi di destinazione

senza modifiche.

Esempio

Nell'esempio seguente viene presentato uno spazio dei nomi di origine e destinazione, ciascuno con annotazioni ed etichette diverse. È possibile visualizzare lo stato dello spazio dei nomi di destinazione prima e dopo l'operazione e il modo in cui le annotazioni e le etichette vengono combinate o sovrascritte nello spazio dei nomi di destinazione.

Prima dell'operazione di ripristino o failover

La tabella seguente illustra lo stato degli spazi dei nomi di origine e di destinazione di esempio prima dell'operazione di ripristino o failover:

Namespace	Annotazioni	Etichette
Namespace ns-1 (origine)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• ambiente=produzione • conformità=hipaa • name=ns-1
Namespace ns-2 (destinazione)	• annotation.one/key: "true" • annotation.three/key: "false"	• ruolo=database

Dopo l'operazione di ripristino

La tabella seguente illustra lo stato dello spazio dei nomi di destinazione di esempio dopo l'operazione di ripristino o failover. Alcune chiavi sono state aggiunte, altre sono state sovrascritte e l' `name`etichetta è stata aggiornata per corrispondere allo spazio dei nomi di destinazione:

Namespace	Annotazioni	Etichette
Namespace ns-2 (destinazione)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • conformità=hipaa • ambiente=produzione • ruolo=database

Campi supportati

Questa sezione descrive i campi aggiuntivi disponibili per le operazioni di ripristino.

Mappatura delle classi di archiviazione

IL `spec.storageClassMapping` L'attributo definisce una mappatura da una classe di archiviazione presente nell'applicazione di origine a una nuova classe di archiviazione nel cluster di destinazione. È possibile utilizzarlo durante la migrazione di applicazioni tra cluster con classi di archiviazione diverse o quando si modifica il backend di archiviazione per le operazioni BackupRestore.

Esempio:

```
storageClassMapping:
  - destination: "destinationStorageClass1"
```

```
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

Annotazioni supportate

Questa sezione elenca le annotazioni supportate per la configurazione di vari comportamenti nel sistema. Se un'annotazione non viene impostata esplicitamente dall'utente, il sistema utilizzerà il valore predefinito.

Annotazione	Tipo	Descrizione	Valore predefinito
proteggi.trident.netapp.io/data-mover-timeout-sec	stringa	Tempo massimo (in secondi) consentito per l'interruzione dell'operazione di spostamento dei dati.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	stringa	Limite massimo di dimensione (in megabyte) per la cache dei contenuti di Kopia.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	stringa	Tempo massimo (in secondi) di attesa affinché i nuovi PersistentVolumeClaim (PVC) creati raggiungano il `Bound` fase prima del fallimento delle operazioni. Si applica a tutti i tipi di ripristino CR (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Utilizzare un valore più alto se il backend di archiviazione o il cluster richiedono spesso più tempo.	"1200" (20 minuti)
protect.trident.netapp.io/retain-mac-address	stringa	Per le VM KubeVirt, imposta `"true"` per mantenere l'indirizzo MAC della prima interfaccia di rete durante il ripristino, ad esempio quando una licenza applicativa è associata a esso. Si applica a tutti i tipi di CR di ripristino e a AppMirrorRelationship (impostalo prima di promuovere la relazione). Per impostazione predefinita, Trident Protect rimuove questo indirizzo MAC per evitare conflitti. Se la VM di origine è ancora in esecuzione sulla stessa rete, arrestala o ripristinala prima in una rete isolata per evitare un indirizzo MAC duplicato.	Non impostato (l'indirizzo MAC è stato rimosso)

Modifica le risorse durante il ripristino utilizzando risorse personalizzate

Le trasformazioni delle risorse ti permettono di modificare una risorsa mentre viene ripristinata. Questo è utile quando la versione ripristinata deve essere diversa dall'originale, ad esempio aggiornando un hostname in una ConfigMap quando ripristini un'applicazione in un cluster diverso.

Come funziona la modifica delle risorse

Il `transformations` campo in `SnapshotRestore`, `BackupRestore`, `AppMirrorRelationship` e altre

risorse di ripristino ti permette di modificare le risorse Kubernetes durante il processo di ripristino. Questo è utile quando l'applicazione ripristinata deve essere configurata in modo diverso dall'originale, per esempio cambiando i nomi host, gli URL del registro, i limiti delle risorse o le variabili d'ambiente.

Le trasformazioni delle risorse utilizzano "RFC 6902 JSON Patch" operazioni e "RFC 6901 JSON Pointer" percorsi per individuare e modificare campi specifici all'interno delle risorse Kubernetes.

Di seguito puoi vedere la struttura di base di un oggetto di ripristino che contiene trasformazioni di risorse:

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: my-restore
  namespace: target-namespace
spec:
  appVaultRef: my-vault
  appArchivePath: /path/to/snapshot
  namespaceMapping:
    - source: source-ns
      destination: target-ns
  transformations:
    - resource:
        kind: ConfigMap       # Required: resource kind
        group: ""             # Optional: API group (empty for core
resources)
        version: ""           # Optional: API version
        name: ""              # Optional: specific resource name
      operations:
        - op: replace         # Operation type
          path: "/data/key"   # JSON Pointer path
          value: "new-value"  # New value (for add/replace/test)
```

Risorse supportate

È possibile utilizzare le trasformazioni delle risorse con risorse che corrispondono ai seguenti criteri:

- **tipo** (obbligatorio): Il tipo di risorsa Kubernetes (ad esempio, `ConfigMap`, `Deployment`, `Pod`)
- **gruppo** (facoltativo): Il gruppo API (ad esempio, `apps`, `route.openshift.io`) — ometti per le risorse core
- **versione** (facoltativo): La versione dell'API (ad esempio, `v1`, `v1beta1`)
- **nome** (facoltativo): Applica solo a una risorsa specifica per nome

NOTA	La modifica di PersistentVolumeClaims e degli spazi dei nomi non è attualmente supportata.

Operazioni supportate

Puoi usare le seguenti operazioni per modificare le risorse: `add`, `copy`, `move`, `remove`, `replace` e `test`.

Aggiungere un valore a una risorsa

Utilizza l'operazione `add` per aggiungere un nuovo campo o valore al percorso specificato. Puoi aggiungere dati a oggetti o array.

Utilizzare un CR

L'esempio seguente aggiunge un selettore di nodi a una `Deployment` risorsa:

```
transformations:
  - resource:
      kind: Deployment
    operations:
      - op: add
        path: "/spec/template/spec/nodeSelector"
        value:
          "topology.kubernetes.io/zone": "us-east-1a"
          disktype: "ssd"
```

Utilizzare la CLI

Utilizzare il seguente comando per eseguire questa trasformazione tramite riga di comando:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
'apps,v1,Deployment:add{"path":"/spec/template/spec/nodeSelector","valu
e":{"topology.kubernetes.io/zone":"us-east-1a","disktype":"ssd"}}'
```

Copia un valore all'interno di una risorsa

Utilizzare l'operazione `copy` per copiare un valore da un percorso a un altro all'interno della stessa risorsa. L'origine non viene modificata.

Utilizzare un CR

Il seguente esempio duplica una chiave dati per un oggetto `ConfigMap`:

```
transformations:
  - resource:
      kind: ConfigMap
```

```
    operations:
      - op: copy
        from: "/data/source-key"
        path: "/data/backup-key"
```

Utilizzare la CLI

Utilizzare il seguente comando per eseguire questa trasformazione tramite riga di comando:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation ',v1,ConfigMap:copy{"from":"/data/source-
key","path":"/data/backup-key"}'
```

Spostare un valore all'interno di una risorsa

Utilizzare l'operazione `move` per spostare un valore da un percorso a un altro all'interno della stessa risorsa. L'origine viene rimossa e il valore viene inserito nella destinazione.

Utilizzare un CR

Il seguente esempio rinomina una chiave dati per un oggetto `ConfigMap`:

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: move
        from: "/data/OLD_KEY"
        path: "/data/NEW_KEY"
```

Utilizzare la CLI

Utilizzare il seguente comando per eseguire questa trasformazione tramite riga di comando:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:move{"from":"/data/OLD_KEY","path":"/data/NEW_KEY"}'
```

Rimuovere un valore da una risorsa

Utilizzare l'operazione `remove` per rimuovere un campo o un valore nel percorso specificato.

Utilizzare un CR

Il seguente esempio rimuove un'annotazione da una risorsa `ConfigMap`:

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: remove
        path: "/metadata/annotations/kubectl.kubernetes.io~1last-
applied-configuration"
```

NOTA Nel percorso per l'esempio precedente, `~1` è la sequenza di escape del puntatore JSON per `/`.

Utilizzare la CLI

Utilizzare il seguente comando per eseguire questa trasformazione tramite riga di comando:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:remove{"path":"/metadata/annotations/kubectl.kubernetes.
io~1last-applied-configuration"}'
```

Sostituire un valore all'interno di una risorsa

Utilizzare l' `replace`operazione per sostituire un valore esistente all'interno di una risorsa al percorso specificato. Il percorso JSON deve già esistere.

Utilizzare un CR

Il seguente esempio cambia il nome host di un `Route` oggetto:

```
transformations:
  - resource:
      kind: Route
      group: route.openshift.io
    operations:
      - op: replace
```

```
        path: "/spec/host"
        value: "prod.example.com"
```

Utilizzare la CLI

Utilizzare il seguente comando per eseguire questa trasformazione tramite riga di comando:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
'route.openshift.io,v1,Route:replace{"path":"/spec/host","value":"prod.
example.com"}'
```

Verifica la modifica della risorsa

Usa l'operazione `test` per verificare che il valore in un percorso corrisponda a quello previsto prima che vengano applicate le altre operazioni. Se il test fallisce, l'intera modifica viene annullata.

Utilizzare un CR

Il seguente esempio aggiorna `database-host` solo se `environment` è impostato su `staging`:

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: test
        path: "/data/environment"
        value: "staging"
      - op: replace
        path: "/data/database-host"
        value: "prod-db.example.com"
```

Utilizzare la CLI

Utilizzare il seguente comando per eseguire questa trasformazione tramite riga di comando:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:test{"path":"/data/environment","value":"staging"},repla
ce{"path":"/data/database-host","value":"prod-db.example.com"}'
```

Replica delle applicazioni con NetApp SnapMirror

Configura e gestisci la replica delle applicazioni utilizzando Trident Protect

Utilizzando Trident Protect, è possibile sfruttare le funzionalità di replica asincrona della tecnologia NetApp SnapMirror per replicare le modifiche dei dati e delle applicazioni da un backend di storage all'altro, sullo stesso cluster o tra cluster diversi.

Annotazioni ed etichette del namespace durante le operazioni di ripristino e failover

Durante le operazioni di ripristino e failover, vengono applicate etichette e annotazioni nel namespace di destinazione in modo che corrispondano alle etichette e alle annotazioni nel namespace di origine. Vengono aggiunte etichette o annotazioni dallo spazio dei nomi di origine che non esistono nello spazio dei nomi di destinazione e le etichette o annotazioni già esistenti vengono sovrascritte per corrispondere al valore dello spazio dei nomi di origine. Le etichette o le annotazioni presenti solo nello spazio dei nomi di destinazione rimangono invariate.

NOTA Se si utilizza Red Hat OpenShift, è importante tenere presente il ruolo fondamentale delle annotazioni dello spazio dei nomi negli ambienti OpenShift. Le annotazioni dello spazio dei nomi garantiscono che i pod ripristinati aderiscano alle autorizzazioni appropriate e alle configurazioni di sicurezza definite dai vincoli del contesto di sicurezza (SCC) di OpenShift e possano accedere ai volumi senza problemi di autorizzazione. Per maggiori informazioni, fare riferimento al"Documentazione dei vincoli del contesto di protezione OpenShift" .

Puoi impedire la sovrascrittura delle annotazioni specifiche nel namespace di destinazione impostando la variabile dell'ambiente Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` prima di eseguire l'operazione di ripristino o failover. Ad esempio:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

NOTA Quando si esegue un'operazione di ripristino o failover, tutte le annotazioni e le etichette dello spazio dei nomi specificate in `restoreSkipNamespaceAnnotations` E `restoreSkipNamespaceLabels` sono esclusi dall'operazione di ripristino o failover. Assicurarsi che queste impostazioni siano configurate durante l'installazione iniziale di Helm. Per saperne di più, fare riferimento a "Configurare le impostazioni aggiuntive del grafico del timone Trident Protect".

Se hai installato l'applicazione sorgente utilizzando Helm con `--create-namespace` bandiera, un trattamento speciale è riservato al `name` etichetta chiave. Durante il processo di ripristino o failover, Trident Protect copia questa etichetta nello spazio dei nomi di destinazione, ma aggiorna il valore al valore dello spazio dei nomi di destinazione se il valore dell'origine corrisponde allo spazio dei nomi di origine. Se questo valore non corrisponde allo spazio dei nomi di origine, viene copiato nello spazio dei nomi di destinazione senza modifiche.

Esempio

Nell'esempio seguente viene presentato uno spazio dei nomi di origine e destinazione, ciascuno con annotazioni ed etichette diverse. È possibile visualizzare lo stato dello spazio dei nomi di destinazione prima e dopo l'operazione e il modo in cui le annotazioni e le etichette vengono combinate o sovrascritte nello spazio dei nomi di destinazione.

Prima dell'operazione di ripristino o failover

La tabella seguente illustra lo stato degli spazi dei nomi di origine e di destinazione di esempio prima dell'operazione di ripristino o failover:

Namespace	Annotazioni	Etichette
Namespace ns-1 (origine)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• ambiente=produzione • conformità=hipaa • name=ns-1
Namespace ns-2 (destinazione)	• annotation.one/key: "true" • annotation.three/key: "false"	• ruolo=database

Dopo l'operazione di ripristino

La tabella seguente illustra lo stato dello spazio dei nomi di destinazione di esempio dopo l'operazione di ripristino o failover. Alcune chiavi sono state aggiunte, altre sono state sovrascritte e l' `name`etichetta è stata aggiornata per corrispondere allo spazio dei nomi di destinazione:

Namespace	Annotazioni	Etichette
Namespace ns-2 (destinazione)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • conformità=hipaa • ambiente=produzione • ruolo=database

NOTA È possibile configurare Trident Protect in modo che blocchi e sblocchi i file system durante le operazioni di protezione dei dati. "Scopri di più sulla configurazione del congelamento del file system con Trident Protect".

Hook di esecuzione durante le operazioni di failover e reverse

Quando si utilizza la relazione AppMirror per proteggere l'applicazione, ci sono comportamenti specifici relativi agli hook di esecuzione di cui è necessario essere a conoscenza durante le operazioni di failover e reverse.

- Durante il failover, gli hook di esecuzione vengono copiati automaticamente dal cluster di origine a quello di destinazione. Non è necessario ricrearli manualmente. Dopo il failover, gli hook di esecuzione sono presenti nell'applicazione e verranno eseguiti durante qualsiasi azione rilevante.
- Durante l'inversione o la risincronizzazione inversa, tutti gli hook di esecuzione esistenti sull'applicazione vengono rimossi. Quando l'applicazione di origine diventa l'applicazione di destinazione, questi hook di esecuzione non sono più validi e vengono eliminati per impedirne l'esecuzione.

Per saperne di più sugli hook di esecuzione, fare riferimento a"Gestire gli hook di esecuzione Trident Protect" .

Impostare una relazione di replica

L'impostazione di una relazione di replica comporta quanto segue:

- Scegliere la frequenza con cui si desidera che Trident Protect esegua uno snapshot dell'app (che include le risorse Kubernetes dell'app e gli snapshot del volume per ciascuno dei volumi dell'app)
- Scelta del programma di replica (include risorse Kubernetes nonché dati dei volumi persistenti)
- Impostazione dell'ora in cui eseguire l'istantanea

Fasi

1. Nel cluster di origine, creare un AppVault per l'applicazione di origine. A seconda del provider di storage, modificare un esempio in "Risorse personalizzate AppVault" per adattare il proprio ambiente:

   Creare un AppVault utilizzando una CR

   a. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-appvault-primary-source.yaml`).

   b. Configurare i seguenti attributi:

      - **metadata.name**: (*required*) il nome della risorsa personalizzata AppVault. Prendere nota del nome scelto, poiché altri file CR necessari per una relazione di replica fanno riferimento a questo valore.
      - **spec.providerConfig**: (*required*) Memorizza la configurazione necessaria per accedere ad AppVault utilizzando il provider specificato. Scegli un bucketName e tutti gli altri dettagli necessari per il tuo provider. Prendere nota dei valori scelti, poiché altri file CR necessari per una relazione di replica fanno riferimento a questi valori. Fare riferimento a "Risorse personalizzate AppVault" per esempi di CRS AppVault con altri provider.
      - **spec.providerCredentials**: (*required*) archivia i riferimenti a qualsiasi credenziale richiesta per accedere ad AppVault utilizzando il provider specificato.
        - **spec.providerCredentials.valueFromSecret**: (*required*) indica che il valore della credenziale deve provenire da un segreto.
          - **Key**: (*required*) la chiave valida del segreto da selezionare.
          - **Nome**: (*obbligatorio*) Nome del segreto che contiene il valore per questo campo. Deve trovarsi nello stesso spazio dei nomi.
        - **spec.providerCredentials.secretAccessKey**: (*required*) la chiave di accesso utilizzata per accedere al provider. Il **nome** deve corrispondere a **spec.providerCredentials.valueFromSecret.name**.
      - **spec.providerType**: (*required*) determina cosa fornisce il backup; ad esempio, NetApp ONTAP S3, S3 generico, Google Cloud o Microsoft Azure. Valori possibili:
        - aws
        - azure
        - gcp
        - generico-s3
        - ONTAP-s3
        - StorageGRID-s3

   c. Dopo aver popolato il `trident-protect-appvault-primary-source.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n
trident-protect
```

Creare un AppVault utilizzando la CLI

a. Creare AppVault, sostituendo i valori tra parentesi con le informazioni dell'ambiente:

```
tridentctl-protect create vault Azure <vault-name> --account
<account-name> --bucket <bucket-name> --secret <secret-name> -n
trident-protect
```

2. Nel cluster di origine, creare l'applicazione di origine CR:

Creare l'applicazione di origine utilizzando una CR

a. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-app-source.yaml`).

b. Configurare i seguenti attributi:

- **metadata.name**: (*required*) il nome della risorsa personalizzata dell'applicazione. Prendere nota del nome scelto, poiché altri file CR necessari per una relazione di replica fanno riferimento a questo valore.
- **spec.includedNamespaces**: (*required*) un array di spazi dei nomi e di etichette associate. Utilizzare i nomi degli spazi dei nomi e, facoltativamente, restringere l'ambito degli spazi dei nomi con le etichette per specificare le risorse esistenti negli spazi dei nomi elencati di seguito. Lo spazio dei nomi dell'applicazione deve far parte di questo array.

  Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: my-app-namespace
      labelSelector: {}
```

c. Dopo aver popolato il `trident-protect-app-source.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-
```

```
namespace
```

Creare l'applicazione di origine utilizzando l'interfaccia CLI

a. Creare l'applicazione di origine. Ad esempio:

```
tridentctl-protect create app <my-app-name> --namespaces
<namespaces-to-be-included> -n <my-app-namespace>
```

3. Facoltativamente, sul cluster di origine, eseguire uno snapshot dell'applicazione di origine. Questo snapshot viene utilizzato come base per l'applicazione sul cluster di destinazione. Se si salta questo passaggio, sarà necessario attendere l'esecuzione del prossimo snapshot pianificato per avere uno snapshot recente. Per creare uno snapshot on-demand, fare riferimento a "Crea un'istantanea on-demand".
4. Nel cluster di origine, creare la pianificazione della replica CR:

Oltre alla pianificazione fornita di seguito, si consiglia di creare una pianificazione separata per gli snapshot giornalieri con un periodo di conservazione di 7 giorni per mantenere uno snapshot comune tra i cluster ONTAP peer. Ciò garantisce che gli snapshot siano disponibili fino a 7 giorni, ma il periodo di conservazione può essere personalizzato in base alle esigenze dell'utente.

NOTA

In caso di failover, il sistema può utilizzare questi snapshot per un massimo di 7 giorni per le operazioni di reverse. Questo approccio rende il processo di reverse più rapido ed efficiente, poiché verranno trasferite solo le modifiche apportate dall'ultimo snapshot, non tutti i dati.

Se una pianificazione esistente per l'applicazione soddisfa già i requisiti di conservazione desiderati, non sono necessarie pianificazioni aggiuntive.

Creare la pianificazione della replica utilizzando un CR

a. Creare una pianificazione di replica per l'applicazione di origine:

i. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-schedule.yaml`).

ii. Configurare i seguenti attributi:

- **metadata.name**: (*required*) il nome della risorsa personalizzata di pianificazione.
- **spec.appVaultRef**: (*Obbligatorio*) Questo valore deve corrispondere al campo metadata.name dell'AppVault per l'applicazione di origine.
- **spec.applicationRef**: (*Obbligatorio*) Questo valore deve corrispondere al campo metadata.name del CR dell'applicazione di origine.
- **Spec.backupRetention**: (*required*) questo campo è obbligatorio e il valore deve essere impostato su 0.
- **Spec.Enabled**: Deve essere impostato su true.
- **spec.granularity**: deve essere impostato su `Custom`.

- **Spec.recurrenceRule**: Consente di definire una data di inizio nell'ora UTC e un intervallo di ricorrenza.
- **Spec.snapshotRetention**: Deve essere impostato su 2.

Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule
  namespace: my-app-namespace
spec:
  appVaultRef: my-appvault-name
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: Custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

i. Dopo aver popolato il `trident-protect-schedule.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-
namespace
```

Creare la pianificazione della replica utilizzando la CLI

a. Crea la pianificazione della replica, sostituendo i valori tra parentesi con le informazioni provenienti dal tuo ambiente:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule <rule> --snapshot-retention
<snapshot_retention_count> -n <my_app_namespace>
```

Esempio:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule  "DTSTART:20220101T000200Z
\nRRULE:FREQ=MINUTELY;INTERVAL=5" --snapshot-retention 2 -n
```

```
<my_app_namespace>
```

5. Nel cluster di destinazione, creare un'applicazione di origine AppVault CR identica a quella AppVault CR applicata al cluster di origine e assegnargli un nome (ad esempio, `trident-protect-appvault-primary-destination.yaml`).
6. Applicare la CR:

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n trident-protect
```

7. Creare una destinazione AppVault CR per l'applicazione di destinazione sul cluster di destinazione. A seconda del provider di storage, modificare un esempio in "Risorse personalizzate AppVault" per adattare il proprio ambiente:
   a. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-appvault-secondary-destination.yaml`).
   b. Configurare i seguenti attributi:
      - **metadata.name**: (*required*) il nome della risorsa personalizzata AppVault. Prendere nota del nome scelto, poiché altri file CR necessari per una relazione di replica fanno riferimento a questo valore.
      - **spec.providerConfig**: (*required*) Memorizza la configurazione necessaria per accedere ad AppVault utilizzando il provider specificato. Scegliere una `bucketName` e tutte le altre informazioni necessarie per il provider. Prendere nota dei valori scelti, poiché altri file CR necessari per una relazione di replica fanno riferimento a questi valori. Fare riferimento a "Risorse personalizzate AppVault" per esempi di CRS AppVault con altri provider.
      - **spec.providerCredentials**: (*required*) archivia i riferimenti a qualsiasi credenziale richiesta per accedere ad AppVault utilizzando il provider specificato.
        - **spec.providerCredentials.valueFromSecret**: (*required*) indica che il valore della credenziale deve provenire da un segreto.
          - **Key**: (*required*) la chiave valida del segreto da selezionare.
          - **Nome**: (*obbligatorio*) Nome del segreto che contiene il valore per questo campo. Deve trovarsi nello stesso spazio dei nomi.
        - **spec.providerCredentials.secretAccessKey**: (*required*) la chiave di accesso utilizzata per accedere al provider. Il **nome** deve corrispondere a **spec.providerCredentials.valueFromSecret.name**.
      - **spec.providerType**: (*required*) determina cosa fornisce il backup; ad esempio, NetApp ONTAP S3, S3 generico, Google Cloud o Microsoft Azure. Valori possibili:
        - aws
        - azure
        - gcp
        - generico-s3
        - ONTAP-s3
        - StorageGRID-s3
   c. Dopo aver popolato il `trident-protect-appvault-secondary-destination.yaml` file con i

valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml
-n trident-protect
```

8. Nel cluster di destinazione, creare un file CR AppMirrorRelationship.

NOTA	Quando si utilizza una CR, creare manualmente lo spazio dei nomi di destinazione prima di applicare la CR. Trident Protect crea automaticamente gli spazi dei nomi solo quando si utilizza la CLI.

Creare una relazione AppMirrorRelationship utilizzando una CR

a. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-relationship.yaml`).

b. Configurare i seguenti attributi:

- **metadata.name:** (obbligatorio) il nome della risorsa personalizzata AppMirrorRelationship.
- **metadata.annotations.protect.trident.netapp.io/retain-mac-address**: (*Opzionale*) Imposta su `"true"` per mantenere l'indirizzo MAC delle VM KubeVirt sul cluster di destinazione. Fai riferimento a "Utilizza le impostazioni di ripristino avanzate Trident Protect".
- **spec.destinationAppVaultRef**: (*required*) questo valore deve corrispondere al nome dell'AppVault per l'applicazione di destinazione sul cluster di destinazione.
- **spec.namespaceMapping**: (*required*) gli spazi dei nomi di destinazione e di origine devono corrispondere allo spazio dei nomi dell'applicazione definito nella rispettiva CR dell'applicazione.
- **Spec.sourceAppVaultRef**: (*required*) questo valore deve corrispondere al nome dell'AppVault per l'applicazione di origine.
- **Spec.sourceApplicationName**: (*required*) questo valore deve corrispondere al nome dell'applicazione di origine definita nell'applicazione di origine CR.
- **spec.sourceApplicationUID**: (obbligatorio) Questo valore deve corrispondere all'UID dell'applicazione sorgente definita nel CR dell'applicazione sorgente.
- **spec.storageClassName**: (*Facoltativo*) Scegli il nome di una classe di archiviazione valida sul cluster. La classe di archiviazione deve essere collegata a una VM di archiviazione ONTAP collegata in peering con l'ambiente di origine. Se non viene specificata la classe di archiviazione, verrà utilizzata per impostazione predefinita la classe di archiviazione predefinita sul cluster.
- **spec.storageClassMapping**: (*Opzionale*) Mappa le classi di archiviazione di origine alle classi di archiviazione di destinazione per i singoli PVC all'interno dello stesso AMR. Ogni voce richiede un `source` e un `destination` nome di classe di archiviazione. Quando impostato, `storageClassMapping` sovrascrive completamente `storageClassName`; i PVC senza una voce di origine corrispondente utilizzano la classe di archiviazione predefinita del cluster, non `storageClassName`.

  Quando usi `storageClassMapping` in una CR AppMirrorRelationship, tieni presente le seguenti restrizioni:

- `storageClassMapping` sovrascrive `storageClassName` completamente. Quando entrambi i campi sono impostati, `storageClassName` viene ignorato.
- I PVC con classi di archiviazione di origine che non hanno una voce corrispondente in `storageClassMapping` usano la classe di archiviazione predefinita del cluster, non `storageClassName`.
- Ogni voce richiede sia un valore `source` che `destination`.
- La modifica della storage class di destinazione può forzare l'eliminazione e la ricreazione dei PVC di destinazione esistenti.

- **Spec.recurrenceRule**: Consente di definire una data di inizio nell'ora UTC e un intervallo di ricorrenza.

  Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: amr-16061e80-1b05-4e80-9d26-d326dc1953d8
  namespace: my-app-namespace
spec:
  desiredState: Established
  destinationAppVaultRef: generic-s3-trident-protect-dst-bucket-
8fe0b902-f369-4317-93d1-ad7f2edc02b5
  namespaceMapping:
    - destination: my-app-namespace
      source: my-app-namespace
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  sourceAppVaultRef: generic-s3-trident-protect-src-bucket-
b643cc50-0429-4ad5-971f-ac4a83621922
  sourceApplicationName: my-app-name
  sourceApplicationUID: 7498d32c-328e-4ddd-9029-122540866aeb
  storageClassName: sc-vsim-2
```

c. Dopo aver popolato il `trident-protect-relationship.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-
namespace
```

Creare un AppMirrorRelationship utilizzando l'interfaccia CLI

a. Crea e applica l'oggetto AppMirrorRelationship, sostituendo i valori tra parentesi con le informazioni provenienti dal tuo ambiente:

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirorrelationship> --destination-app-vault
<my_vault_name> --source-app-vault <my_vault_name> --recurrence
-rule <rule> --namespace-mapping <ns_mapping> --source-app-id
<source_app_UID> --source-app <my_source_app_name> --storage
-class <storage_class_name> --storageclass-mapping
<src_class:dst_class> -n <application_namespace>
```

NOTA Usa `--storageclass-mapping` per mappare singole classi di archiviazione di origine su classi di archiviazione di destinazione. Specifica una o più coppie `source:destination` separate da virgole (ad esempio, `"class-a:class-b,class-c:class-d"`). Usa `*:destination-class` come carattere jolly per mappare tutti i PVC non mappati a una singola classe di archiviazione di destinazione. Quando `--storageclass-mapping` è impostato, `--storage-class` viene ignorato.

Esempio con una singola classe di archiviazione di destinazione (`--storage-class`):

```
tridentctl-protect create appmirrorrelationship my-amr
--destination-app-vault appvault2 --source-app-vault appvault1
--recurrence-rule
"DTSTART:20220101T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5"
--source-app my-app --namespace-mapping "my-source-ns1:my-dest-
ns1,my-source-ns2:my-dest-ns2" --source-app-id 373f24c1-5769-
404c-93c3-5538af6ccc36 --storage-class my-storage-class -n my-
dest-ns1
```

9. (*Optional*) nel cluster di destinazione, verificare lo stato e lo stato della relazione di replica:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Failover sul cluster di destinazione

Utilizzando Trident Protect è possibile eseguire il failover delle applicazioni replicate su un cluster di destinazione. Questa procedura interrompe la relazione di replica e porta l'app online sul cluster di destinazione. Trident Protect non arresta l'app sul cluster di origine se era operativa.

Questa procedura esegue il failover all'istantanea più recente. Per eseguire il failover su un'istantanea precedente, consulta "Esegui il failover a uno snapshot precedente usando Trident Protect".

Fasi

1. (*Opzionale*) Per mantenere l'indirizzo MAC delle VM KubeVirt, annota AppMirrorRelationship prima di promuoverlo. Fai riferimento a "Utilizza le impostazioni di ripristino avanzate Trident Protect".

```
kubectl annotate appmirrorrelationship <relationship_name> -n
<application_namespace> protect.trident.netapp.io/retain-mac-address
="true"
```

2. Nel cluster di destinazione, modificare il file CR AppMirrorRelationship (ad esempio, `trident-protect-relationship.yaml`) e modificare il valore di **spec.desiredState** in `Promoted`.
3. Salvare il file CR.
4. Applicare la CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. (*Optional*) creare tutte le pianificazioni di protezione necessarie per l'applicazione in cui è stato eseguito il failover.
6. (*Optional*) controllare lo stato e lo stato della relazione di replica:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Risincronizzazione di una relazione di replica non riuscita

L'operazione di risincronizzazione ristabilisce la relazione di replica. Dopo aver eseguito un'operazione di risincronizzazione, l'applicazione di origine diventa l'applicazione in esecuzione e tutte le modifiche apportate all'applicazione in esecuzione sul cluster di destinazione vengono scartate.

Il processo arresta l'applicazione sul cluster di destinazione prima di ristabilire la replica.

IMPORTANTE	Tutti i dati scritti nell'applicazione di destinazione durante il failover andranno persi.

Fasi

1. Opzionale: Nel cluster di origine, creare uno snapshot dell'applicazione di origine. In questo modo si garantisce che vengano acquisite le ultime modifiche dal cluster di origine.
2. Nel cluster di destinazione, modificare il file CR AppMirrorRelationship (ad esempio, `trident-protect-relationship.yaml`) e modificare il valore di spec.desiredState in `Established`.
3. Salvare il file CR.
4. Applicare la CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. Rimuovere eventuali pianificazioni di protezione sul cluster di destinazione per proteggere l'applicazione in cui è stato eseguito il failover. Qualsiasi pianificazione rimanente causa errori di snapshot dei volumi.

Risincronizzazione inversa di una relazione di replica non riuscita

Quando si esegue la risincronizzazione inversa di una relazione di replica non riuscita, l'applicazione di destinazione diventa l'applicazione di origine e l'origine diventa la destinazione. Le modifiche apportate all'applicazione di destinazione durante il failover vengono mantenute.

Fasi

1. Nel cluster di destinazione originale, eliminare la CR AppMirrorRelationship. Ciò fa sì che la destinazione diventi l'origine. Rimuovere eventuali pianificazioni relative alla protezione sul nuovo cluster di destinazione.
2. Impostare una relazione di replica applicando i file CR utilizzati originariamente per impostare la relazione con i cluster opposti.
3. Assicurarsi che la nuova destinazione (cluster di origine originale) sia configurata con entrambi i CRS AppVault.
4. Impostare una relazione di replica sul cluster opposto, configurando i valori per la direzione inversa.

Invertire la direzione di replica dell'applicazione

Quando si inverte la direzione della replica, Trident Protect sposta l'applicazione sul backend di archiviazione di destinazione, continuando a replicare sul backend di archiviazione di origine. Trident Protect arresta l'applicazione di origine e replica i dati nella destinazione prima di eseguire il failover sull'app di destinazione.

In questa situazione, si sta sostituendo l'origine e la destinazione.

Fasi

1. Nel cluster di origine, creare uno snapshot di arresto:

Creare un'istantanea di arresto utilizzando una CR

a. Disattivare le pianificazioni dei criteri di protezione per l'applicazione di origine.

b. Creare un file ShutdownSnapshot CR:

i. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-shutdownsnapshot.yaml`).

ii. Configurare i seguenti attributi:

- **metadata.name**: (*required*) il nome della risorsa personalizzata.
- **Spec.AppVaultRef**: (*required*) questo valore deve corrispondere al campo metadata.name dell'AppVault per l'applicazione di origine.
- **Spec.ApplicationRef**: (*required*) questo valore deve corrispondere al campo metadata.name del file CR dell'applicazione di origine.

Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ShutdownSnapshot
metadata:
  name: replication-shutdown-snapshot-afc4c564-e700-4b72-86c3-
c08a5dbe844e
```

```
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  applicationRef: my-app-name
```

c. Dopo aver popolato il `trident-protect-shutdownsnapshot.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-
namespace
```

Creare uno snapshot di arresto utilizzando l'interfaccia CLI

a. Creare l'istantanea di arresto, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente. Ad esempio:

```
tridentctl-protect create shutdownsnapshot <my_shutdown_snapshot>
--appvault <my_vault> --app <app_to_snapshot> -n
<application_namespace>
```

2. Sul cluster di origine, dopo il completamento dello snapshot di arresto, ottenere lo stato dello snapshot di arresto:

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. Nel cluster di origine, trovare il valore di **shutdownsnapshot.status.appArchivePath** utilizzando il seguente comando e registrare l'ultima parte del percorso del file (chiamato anche nome di base; questo sarà tutto dopo l'ultima barra):

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o
jsonpath='{.status.appArchivePath}'
```

4. Eseguire un failover dal nuovo cluster di destinazione al nuovo cluster di origine, con la seguente modifica:

NOTA	Nel passaggio 2 della procedura di failover, includere il `spec.promotedSnapshot` campo nel file CR AppMirrorRelationship e impostarne il valore sul nome di base registrato nel passaggio 3 di cui sopra.

5. Eseguire le operazioni di risincronizzazione inversa descritte in Risincronizzazione inversa di una relazione di replica non riuscita.
6. Attiva le pianificazioni della protezione sul nuovo cluster di origine.

Risultato

A causa della replica inversa, si verificano le seguenti azioni:

- Viene acquisita un'istantanea delle risorse Kubernetes dell'applicazione di origine.
- I pod dell'applicazione di origine vengono interrotti correttamente eliminando le risorse Kubernetes dell'applicazione (lasciando PVC e PVS in posizione).
- Una volta spenti i pod, vengono acquisite e replicate le istantanee dei volumi dell'applicazione.
- Le relazioni di SnapMirror vengono interrotte, rendendo i volumi di destinazione pronti per la lettura/scrittura.
- Le risorse Kubernetes dell'applicazione vengono ripristinate dallo snapshot pre-shutdown, utilizzando i dati del volume replicati dopo l'arresto dell'applicazione di origine.
- La replica viene ristabilita in senso inverso.

Eseguire il failback delle applicazioni nel cluster di origine originale

Utilizzando Trident Protect, è possibile ottenere il "fail back" dopo un'operazione di failover utilizzando la seguente sequenza di operazioni. In questo flusso di lavoro per ripristinare la direzione di replicazione originale, Trident Protect replica (risincronizza) tutte le modifiche apportate all'applicazione di origine prima di invertire la direzione di replicazione.

Questo processo inizia da una relazione che ha completato un failover verso una destinazione e prevede i seguenti passaggi:

- Iniziare con uno stato di failover.
- Risincronizzazione inversa della relazione di replica.

AVVERTENZA	Non eseguire una normale operazione di risincronizzazione, in quanto i dati scritti nel cluster di destinazione verranno eliminati durante la procedura di failover.

- Invertire la direzione di replica.

Fasi

1. Eseguire i Risincronizzazione inversa di una relazione di replica non riuscitapassaggi.
2. Eseguire i Invertire la direzione di replica dell'applicazionepassaggi.

Eliminare una relazione di replica

È possibile eliminare una relazione di replica in qualsiasi momento. Quando si elimina la relazione di replica dell'applicazione, vengono generate due applicazioni separate senza alcuna relazione tra di esse.

Fasi

1. Nel cluster di desinazione corrente, eliminare AppMirrorRelationship CR:

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

Esegui il failover a uno snapshot precedente usando Trident Protect

Usa Trident Protect per eseguire il failover di un'applicazione replicata verso uno snapshot precedente, consentendoti di ripristinare i dati a un momento precedente al loro danneggiamento o alla loro compromissione. Per impostazione predefinita, il failover utilizza lo snapshot trasferito più di recente.

AVVERTENZA	Il passaggio a uno snapshot precedente fa sì che ONTAP SnapMirror elimini gli snapshot sul cluster di destinazione più recenti dello snapshot selezionato, perché non sono più validi. Aggiungi l'annotazione in questa procedura solo se questa perdita di dati è accettabile.

Prima di iniziare

- Esiste un oggetto AppMirrorRelationship ed è nello stato `Established`. Per crearne uno, consulta "Impostare una relazione di replica". Per verificare lo stato, esegui `kubectl get amr -n <namespace> <relationship_name> -o=jsonpath='{.status}'`.
- I cluster di origine e di destinazione utilizzano backend di storage ONTAP con replica SnapMirror. Azure NetApp Files e altri backend non-ONTAP non sono supportati. Consulta "Requisiti per la replica SnapMirror".
- Nell'applicazione di origine esiste una pianificazione di snapshot (con `snapshotRetention` valore maggiore di 1).

Identifica l'istantanea

Ogni istantanea ha un orario di completamento, quindi puoi scegliere il momento che desideri.

Fasi

1. Sul cluster di origine, elenca gli snapshot:

```
kubectl get snapshots -n my-app-namespace
```

2. Individua l'istantanea a cui eseguire il failover. Utilizza il `completionTimestamp` valore per selezionare il momento desiderato.
3. Ottieni il percorso dell'archivio di quella snapshot:

```
kubectl get snapshots <snapshot_name> -n my-app-namespace -o jsonpath
='{.status.appArchivePath}'
```

4. Registra l'ultima parte del percorso, ovvero tutto ciò che segue l'ultima barra. Questa parte è il *basename*. Ad esempio:

```
20231114173500_schedule-20231114173500_f8df8bd3-a727-4a14-b954-
9666aa618b55
```

NOTA	Usa il nome base, non il nome della risorsa personalizzata snapshot.

Risultato

Il nome di base dello snapshot selezionato viene registrato.

Passa allo snapshot selezionato in caso di errore

Specifica l'istantanea da promuovere e conferma la perdita di dati aggiungendo un'annotazione.

Fasi

1. Sul cluster di destinazione, aggiungi l'annotazione a AppMirrorRelationship. Senza l'annotazione, un failover può utilizzare solo lo snapshot trasferito più di recente.

```
kubectl annotate appmirrorrelationship <relationship_name> -n my-app-
namespace protect.trident.netapp.io/allow-promotion-with-older-
snapshots="true"
```

2. Modifica il file CR AppMirrorRelationship (ad esempio `trident-protect-relationship.yaml`):

   - Imposta **spec.desiredState** su `Promoted`.
   - Imposta **spec.promotedSnapshot** sul nome base che hai registrato in Identifica l'istantanea.

   Ad esempio:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: <relationship_name>
  namespace: <namespace>
  annotations:
    protect.trident.netapp.io/allow-promotion-with-older-snapshots:
"true"
spec:
  desiredState: Promoted
  promotedSnapshot: <basename>
```

3. Applicare la CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. Verifica lo stato della relazione:

```
kubectl get amr -n my-app-namespace <relationship_name> -o=jsonpath
='{.status}' | jq
```

Risultato

La relazione passa allo `Promoted` stato e l'applicazione viene eseguita sul cluster di destinazione con i dati dello snapshot selezionato.

Se la relazione rimane nello `Promoting`stato e `status.error`mostra `The snapshot provided is invalid or older than the most recently transferred snapshot`, l'annotazione è mancante o non è impostata su `true`. Aggiungi l'annotazione e riprova.

Quali sono le prossime novità?

Per tornare al cluster originale, esegui un "risincronizzazione inversa" e poi "invertire la direzione della replica".

AVVERTENZA	Per tornare al cluster originale, non eseguire una normale risincronizzazione. Una normale risincronizzazione scarta l'applicazione di failover e ripristina la relazione con l'istantanea di origine più recente.

Informazioni correlate

- "Configura e gestisci la replica delle applicazioni utilizzando Trident Protect"
- "Failover sul cluster di destinazione"
- "Risincronizzazione di una relazione di replica non riuscita"
- "Eseguire il failback delle applicazioni nel cluster di origine originale"
- "Creare un piano di data Protection"

Migrare le applicazioni utilizzando Trident Protect

È possibile migrare le applicazioni tra cluster o in classi di archiviazione diverse ripristinando i dati di backup.

NOTA	Quando si esegue la migrazione di un'applicazione, tutti i collegamenti di esecuzione configurati per l'applicazione vengono migrati con l'applicazione. Se è presente un gancio di esecuzione post-ripristino, viene eseguito automaticamente come parte dell'operazione di ripristino.

Operazioni di backup e ripristino

Per eseguire operazioni di backup e ripristino per i seguenti scenari, è possibile automatizzare attività di backup e ripristino specifiche.

Clona nello stesso cluster

Per clonare un'applicazione nello stesso cluster, crea una snapshot o un backup e ripristina i dati nello stesso cluster.

Fasi

1. Effettuare una delle seguenti operazioni:
   a. "Creare un'istantanea".
   b. "Creare un backup".
2. Nello stesso cluster, eseguire una delle seguenti operazioni, a seconda che sia stato creato uno snapshot o un backup:

a. "Ripristinare i dati dalla snapshot".

b. "Ripristinare i dati dal backup".

Clona in un cluster diverso

Per clonare un'applicazione su un cluster diverso (eseguire un clone tra cluster), creare un backup sul cluster di origine, quindi ripristinare il backup su un cluster diverso. Assicurarsi che Trident Protect sia installato sul cluster di destinazione.

NOTA	È possibile replicare un'applicazione tra cluster diversi utilizzando "Replica SnapMirror".

Fasi

1. "Creare un backup".
2. Verificare che AppVault CR per il bucket di storage a oggetti che contiene il backup sia stato configurato sul cluster di destinazione.
3. Sul cluster di destinazione, "ripristinare i dati dal backup".

Eseguire la migrazione delle applicazioni da una classe di storage a un'altra

È possibile migrare le applicazioni da una classe di archiviazione a una diversa ripristinando un backup nella classe di archiviazione di destinazione.

Ad esempio (escludendo i segreti dalla CR di ripristino):

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: "${snapshotRestoreCRName}"
spec:
  appArchivePath: "${snapshotArchivePath}"
  appVaultRef: "${appVaultCRName}"
  namespaceMapping:
    - destination: "${destinationNamespace}"
      source: "${sourceNamespace}"
  storageClassMapping:
    - destination: "${destinationStorageClass}"
      source: "${sourceStorageClass}"
  resourceFilter:
    resourceMatchers:
      kind: Secret
      version: v1
    resourceSelectionCriteria: exclude
```

Ripristinare l'istantanea utilizzando una CR

Fasi

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-snapshot-`

`restore-cr.yaml`.

2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono memorizzati i contenuti dello snapshot. Per trovare il percorso, utilizzare il seguente comando:

     ```
     kubectl get snapshots <my-snapshot-name> -n trident-protect -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui sono memorizzati i contenuti dello snapshot.
   - **spec.namespaceMapping**: mappatura dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituire `my-source-namespace` e `my-destination-namespace` con le informazioni del proprio ambiente.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: trident-protect
     spec:
       appArchivePath: my-snapshot-path
       appVaultRef: appvault-name
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. Se si desidera, è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda le risorse contrassegnate con determinate etichette:
   - **ResourceFilter.resourceSelectionCriteria**: (Necessario per il filtraggio) utilizzare `include or exclude` per includere o escludere una risorsa definita in resourceMatcher. Aggiungere i seguenti parametri resourceMatcher per definire le risorse da includere o escludere:
     - **ResourceFilter.resourceMatchers**: Una matrice di oggetti resourceMatcher. Se si definiscono più elementi in questa matrice, questi corrispondono come un'operazione OR e i campi all'interno di ogni elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.
       - **ResourceMatchers[].group**: (*Optional*) Gruppo della risorsa da filtrare.
       - **ResourceMatchers[].Kind**: (*Optional*) tipo di risorsa da filtrare.
       - **ResourceMatchers[].version**: (*Optional*) versione della risorsa da filtrare.
       - **ResourceMatchers[].names**: (*Optional*) nomi nel campo Kubernetes metadata.name della risorsa da filtrare.
       - **ResourceMatchers[].namespaces**: (*Optional*) Namespaces nel campo Kubernetes

metadata.name della risorsa da filtrare.

- **ResourceMatchers[].labelSelectors**: (*Optional*) stringa del selettore di etichette nel campo Kubernetes metadata.name della risorsa come definito nella "Documentazione Kubernetes" . Ad esempio: `"trident.netapp.io/os=linux"`.

  Ad esempio:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Dopo aver popolato il `trident-protect-snapshot-restore-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Ripristinare la snapshot utilizzando la CLI

Fasi

1. Ripristinare lo snapshot in uno spazio dei nomi diverso, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente.

   - L'`snapshot`argomento utilizza uno spazio dei nomi e un nome snapshot nel formato `<namespace>/<name>.
   - L'`namespace-mapping`argomento utilizza spazi dei nomi separati da due punti per mappare gli spazi dei nomi di origine agli spazi dei nomi di destinazione corretti nel formato `source1:dest1,source2:dest2.

   Ad esempio:

```
tridentctl-protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

Proteggi le macchine virtuali KubeVirt con Trident Protect

Trident Protect ti permette di eseguire il backup e il ripristino delle macchine virtuali (VM) KubeVirt in esecuzione su OpenShift Virtualization. Puoi proteggere tutte le VM in uno o più namespace contemporaneamente, oppure usare il campo `includedVirtualMachines` per selezionare VM specifiche per nome. Trident Protect raccoglie automaticamente tutte le risorse VM dipendenti durante il backup e, durante il ripristino, puoi recuperare l'intero archivio o ripristinare selettivamente singole VM.

Modalità di protezione KubeVirt

Trident Protect supporta due modalità per definire quali KubeVirt VM proteggere. Scegli una modalità quando definisci un'applicazione:

Modalità	Campo	Descrizione
Basato sul namespace	`includedNamespaces`	Proteggi tutte le risorse corrispondenti in uno o più namespace. Usa un `labelSelector` sulla voce del namespace per controllare quali VM vengono incluse dinamicamente.
Con ambito VM	`includedVirtualMachines`	Proteggi specifiche KubeVirt VM per namespace e nome. Usa questa modalità quando le VM sono distribuite tra namespace diversi o quando non vuoi proteggere tutte le risorse di un namespace.

Devi specificare esattamente un campo (`includedNamespaces` o `includedVirtualMachines` nel `ApplicationSpec`. Non puoi usare entrambi nella stessa applicazione.

Entrambe le modalità supportano filtri aggiuntivi:

- Usa `resourceFilter` per includere o escludere risorse specifiche.
- Usa `includedClusterScopedResources` per includere risorse compatibili con ambito cluster, come oggetti `StorageClass`.

NOTA

- Trident Protect può bloccare e sbloccare i filesystem delle VM KubeVirt durante le operazioni di protezione dei dati per garantire la coerenza. Per i dettagli di configurazione, vedi "Protezione dei dati con le macchine virtuali KubeVirt".
- Per creare snapshot di tutti i dischi di una VM nello stesso momento, usa gli snapshot di gruppi di volumi. Fai riferimento a "Utilizza gli snapshot dei gruppi di volumi con Trident Protect".

Risorse raccolte durante il backup e lo snapshot

Quando crei un backup o uno snapshot per un'applicazione che include KubeVirt VM, Trident Protect raccoglie le risorse VM selezionate e scopre automaticamente le risorse da cui dipendono.

Il comportamento di raccolta si applica in due casi:

- Applicazioni con ambito VM che usano `includedVirtualMachines`.
- Applicazioni basate su namespace che contengono KubeVirt `VirtualMachine` risorse.

raccolta con ambito VM

Per le applicazioni con ambito VM, Trident Protect inizia con ciascuna `VirtualMachine` elencata in `includedVirtualMachines` e individua le risorse dipendenti necessarie per ripristinare quella VM.

Tipo di risorsa	Ambito	Come viene scoperto
`VirtualMachine`	Namespaced	Corrispondenza diretta dall'elenco `includedVirtualMachines`.
`VirtualMachineInstance`	Namespaced	Stesso namespace e nome della VM, se la VM è in esecuzione.
`VirtualMachineInstancetype`	Namespaced	Citato da `spec.instancetype` quando `kind: VirtualMachineInstancetype`.
`VirtualMachineClusterInstancetype`	Cluster	Riferito da `spec.instancetype` quando `kind` non è impostato o `kind: VirtualMachineClusterInstancetype`.
`VirtualMachinePreference`	Namespaced	Citato da `spec.preference` quando `kind: VirtualMachinePreference`.
`VirtualMachineClusterPreference`	Cluster	Riferito da `spec.preference` quando `kind` non è impostato o `kind: VirtualMachineClusterPreference`.
`DataVolume`	Namespaced	Citato da `spec.template.spec.volumes[].dataVolume.name`.
`PersistentVolumeClaim`	Namespaced	Scoperti dai PVC di supporto di DataVolume, che utilizzano lo stesso nome del DataVolume; riferimenti diretti ai PVC in `persistentVolumeClaim.claimName`; PVC di dump della memoria in `memoryDump.claimName`; e PVC effimeri in `ephemeral.persistentVolumeClaim.claimName`.
`PersistentVolume`	Cluster	Scoperto dal supporto PV per ogni PVC raccolto utilizzando `spec.volumeName`.
`Secret`	Namespaced	Scoperti dai segreti del volume in `secret.secretName`; segreti di estrazione delle immagini containerDisk; riferimenti a segreti di dati utente e dati di rete CloudInitNoCloud e CloudInitConfigDrive; segreti Sysprep; segreti di estrazione delle immagini dei container KernelBoot; e segreti SSH o password AccessCredentials.
`ConfigMap`	Namespaced	Scoperto dal volume ConfigMaps in `configMap.name` e Sysprep ConfigMaps.
`ServiceAccount`	Namespaced	Scoperto da `serviceAccount.serviceAccountName` riferimenti di volume.
`NetworkAttachmentDefinition`	Namespaced	Scoperto dalle reti Multus `spec.template.spec.networks[].multus.networkName` quando il nome della rete non contiene `/`. Vengono raccolti solo i riferimenti NetworkAttachmentDefinition dello stesso namespace. I riferimenti tra namespace diversi vengono ignorati.

Tipo di risorsa	Ambito	Come viene scoperto
`Pod`	Namespaced	Individuato per `virt-launcher`Pod utilizzando il selettore di etichette `kubevirt.io=virt-launcher,vm.kubevirt.io/name=<vmName>`.
Classe di archiviazione, volumi Trident e backend Trident	Cluster	Raccolti nello stesso modo delle applicazioni basate su namespace, in base ai PVC rilevati per la VM.

NOTA Dopo la raccolta di tutte le risorse con ambito VM e namespace, Trident Protect applica la procedura standard `resourceFilter`, se configurata. La raccolta delle risorse con ambito cluster procede quindi normalmente.

Raccolta basata sul namespace

Per le applicazioni basate su namespace, Trident Protect analizza i namespace selezionati e raccoglie tutte le risorse corrispondenti. Se vengono trovati oggetti KubeVirt `VirtualMachine`, Trident Protect raccoglie automaticamente le risorse dipendenti, inclusi i tipi di istanza, le preferenze e `virt-launcher` Pod, nello stesso modo in cui fa per le applicazioni con ambito VM.

Ciò significa che le risorse specifiche della VM sono incluse nel backup o nello snapshot anche quando l'applicazione è definita con `includedNamespaces`.

Proteggi le macchine virtuali usando definizioni di applicazioni basate su namespace

Quando crei un'applicazione basata su namespace, Trident Protect analizza i namespace selezionati e raccoglie tutte le risorse, incluse le KubeVirt VMs. Per ogni virtual machine trovata, Trident Protect raccoglie anche le risorse dipendenti necessarie per eseguire il backup e il ripristino di quella VM.

Esempio di CR di applicazione basata su namespace:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedNamespaces:
    - namespace: <vm_namespace>
```

Dopo aver creato l'applicazione, usa i flussi di lavoro standard di Trident Protect per i backup e gli snapshot:

- Per esempi di snapshot YAML, vedi "Crea un'istantanea on-demand".
- Per esempi di backup YAML, vedi "Crea un backup su richiesta".

Comportamento di ripristino per le applicazioni VM basate su namespace

Poiché le applicazioni basate su namespace raccolgono risorse dai namespace selezionati, il backup o lo snapshot possono includere oggetti runtime KubeVirt e metadati gestiti da KubeVirt. Alcuni di questi oggetti

non dovrebbero essere ripristinati direttamente perché rappresentano lo stato runtime del cluster di origine.

Durante il ripristino, Trident Protect applica trasformazioni compatibili con KubeVirt per rimuovere i metadati del cluster di origine e saltare le risorse KubeVirt temporanee. Questo aiuta KubeVirt a ricreare la VM in modo pulito nell'ambiente ripristinato.

Risorsa	Trasforma
VirtualMachine	Rimuove etichette e annotazioni che iniziano con `kubevirt.io/`.
VirtualMachineInstance	Rimuove etichette e annotazioni che iniziano con `kubevirt.io/`.
VirtualMachineInstanceMigration	Ignorato e non ripristinato.
VirtualMachineSnapshot	Ignorato e non ripristinato.
VirtualMachineSnapshotContent	Ignorato e non ripristinato.
Pod (`virt-launcher`)	Saltato quando il pod ha l'etichetta `kubevirt.io=virt-launcher`.

NOTA

- Trident Protect ripristina le risorse KubeVirt in un ordine definito per evitare problemi di dipendenza. I CDI DataVolumes (`cdi.kubevirt.io` vengono ripristinati per primi così i dati del disco della VM sono disponibili prima che vengano creati gli oggetti VM. Dopodiché, le VirtualMachines (`kubevirt.io` vengono ripristinate prima delle VirtualMachineInstances.
- Per impostazione predefinita, Trident Protect rimuove l'indirizzo MAC dalla prima interfaccia di rete di ogni VM ripristinata. Per mantenerlo, usa l' `protect.trident.netapp.io/retain-mac-address`annotazione. Fai riferimento a "Utilizza le impostazioni di ripristino avanzate Trident Protect".

Protezione dinamica delle VM con selettori di etichette

Se vuoi proteggere le VM in modo dinamico senza elencare ogni nome di VM, usa definizioni di applicazioni basate su namespace con selettori di etichette.

Solo il VirtualMachine CR necessita di etichette. Le risorse dipendenti come DataVolumes, PVC, Secrets e ConfigMaps vengono raccolte automaticamente.

Fasi

1. Etichetta le CR delle macchine virtuali con un'etichetta condivisa:

```
kubectl label virtualmachine <vm_name 1> app=<label_value> -n
<vm_namespace>
kubectl label virtualmachine <vm_name_2> app=<label_value> -n
<vm_namespace>
```

2. Crea un'applicazione CR con `includedNamespaces` e un `labelSelector`:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
```

```
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedNamespaces:
    - namespace: <vm_namespace>
      labelSelector:
        matchLabels:
          app: <label_value>
```

3. Crea backup e ripristini utilizzando il flusso di lavoro standard di Trident Protect.

Ulteriori dettagli:

- Puoi aggiungere o rimuovere macchine virtuali dalla protezione in qualsiasi momento modificando le etichette:

```
kubectl label virtualmachine <vm_name_to_add> app=<label_value> -n
<vm_namespace>
kubectl label virtualmachine <vm_name_to_remove> app
=<different_label_value> -n <vm_namespace>
```

- Puoi anche usare i selettori di etichette su più namespace:

```
spec:
  includedNamespaces:
    - namespace: <vm_namespace_1>
      labelSelector:
        matchLabels:
          <label_key>: <label_value>
    - namespace: <vm_namespace_2>
      labelSelector:
        matchLabels:
          <label_key>: <label_value>
```

Proteggi macchine virtuali specifiche con includedVirtualMachines

Usa `includedVirtualMachines` quando vuoi proteggere un insieme denominato di VirtualMachines attraverso uno o più namespace e ripristinare selettivamente VM specifiche da un archivio.

Sia le CR di applicazione che quelle di ripristino utilizzano lo stesso `includedVirtualMachines` formato di selettore.

Utilizza questo selettore in contesti diversi:

- In una CR dell'applicazione, definisci quali VM sono protette.
- In una richiesta di ripristino (CR), definisci quali macchine virtuali vengono ripristinate dall'archivio di

backup o Snapshot.

NOTA Nei restore CR, `includedVirtualMachines` è facoltativo. Se non includi questo campo, viene ripristinato l'intero archivio. Quando specificato, vengono ripristinate solo le VM elencate e le relative risorse dipendenti.

Il codice YAML riportato di seguito mostra la struttura del selettore condiviso:

```
includedVirtualMachines:
  - namespace: <vm_namespace_1>
    names:
      - <vm_name_1>
      - <vm_name_2>
  - namespace: <vm_namespace_2>
    names:
      - <vm_name_3>
```

NOTA Più voci in `includedVirtualMachines` possono fare riferimento a namespace diversi, il che consente definizioni e ripristini di applicazioni VM-aware su più namespace.

Campo	Tipo	Descrizione
spazio dei nomi	stringa	Spazio dei nomi Kubernetes che contiene gli oggetti della macchina virtuale di destinazione.
nomi	elenco di stringhe	Uno o più nomi di macchine virtuali in quello spazio dei nomi.

Esempio di applicazione basata su macchina virtuale:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedVirtualMachines:
    - namespace: <vm_namespace_1>
      names:
        - <vm_name_1>
        - <vm_name_2>
    - namespace: <vm_namespace_2>
      names:
        - <vm_name_3>
  includedClusterScopedResources:
    - group: storage.k8s.io
      kind: StorageClass
```

Ripristina specifiche macchine virtuali KubeVirt

Puoi usare `includedVirtualMachines` sui CR di ripristino per ripristinare solo le KubeVirt VM selezionate da un backup, uno Snapshot o un archivio di Snapshot replicato.

Quando `includedVirtualMachines` viene specificato su un CR di ripristino, Trident Protect ripristina solo le VirtualMachines elencate e le relative risorse dipendenti, come `VirtualMachineInstance`, `DataVolume`, `PersistentVolumeClaim`, `PersistentVolume`, `Secret`, `ConfigMap`, `ServiceAccount` e altri oggetti correlati alla VM.

Se non specifichi `includedVirtualMachines` nel restore CR, Trident Protect ripristina l'intero archivio secondo la configurazione del restore CR.

CR di ripristino e comandi CLI supportati

La tabella seguente elenca i CR di ripristino che supportano `includedVirtualMachines` e i relativi comandi CLI.

Ripristina CR	Comando CLI	Descrizione
`BackupRestore`	`tridentctl-protect create backuprestore`	Ripristina da un backup a mappature di namespace nuove o diverse. Supporta `namespaceMapping` e `storageClassMapping`.
`BackupInplaceRestore`	`tridentctl-protect create backupinplacerestore`	Esegui il ripristino da un backup nello spazio dei nomi o negli spazi dei nomi originali.
`SnapshotRestore`	`tridentctl-protect create snapshotrestore`	Ripristina da uno Snapshot a mappature di namespace nuove o diverse. Supporta `namespaceMapping` e `storageClassMapping`.
`SnapshotInplaceRestore`	`tridentctl-protect create snapshotinplacerestore`	Esegui il ripristino da una Snapshot nello spazio dei nomi o negli spazi dei nomi originali.
`ReplicateSnapshotRestore`	Solo YAML, oppure usa il comando CLI supportato se disponibile nel tuo ambiente	Ripristina da uno snapshot replicato. Supporta `namespaceMapping`, `storageClassMapping` e il rispristino in-place tramite il flag `inPlaceRestore`.

NOTA La `tridentctl-protect` CLI non ha un flag per `includedVirtualMachines` nei comandi di ripristino. Per ripristinare VM specifiche, devi applicare direttamente un file YAML di restore CR usando `kubectl apply`. Puoi scrivere il file YAML da zero oppure usare la CLI con `--dry-run --output yaml` per generare un file iniziale, aggiungere il campo `includedVirtualMachines` sotto `spec`, e poi applicarlo.

Comportamento di ripristino a livello di macchina virtuale

Il `includedVirtualMachines`campo in una richiesta di ripristino (CR) utilizza lo stesso formato di selettore della richiesta di applicazione (CR):

```
includedVirtualMachines:
  - namespace: <source_vm_namespace>
    names:
      - <vm_name_1>
      - <vm_name_2>
```

Per i CR di ripristino, i valori dello spazio dei nomi `includedVirtualMachines` si riferiscono agli spazi dei nomi di origine così come esistono nel backup, Snapshot o archivio di Snapshot replicato.

Se usi `namespaceMapping`, Trident Protect seleziona prima le VM richieste dall'archivio di origine e poi applica le mappature dello spazio dei nomi durante il ripristino. Se usi `storageClassMapping`, Trident Protect applica le mappature della storage class durante la fase di trasformazione del ripristino.

Genera il file YAML di ripristino con la CLI

Puoi usare `tridentctl-protect` per generare un CR di ripristino di base e poi modificare il file YAML per aggiungere `includedVirtualMachines`.

```
tridentctl-protect create <restore_command> <restore_name> \
  <restore_options> \
  --dry-run --output yaml > <restore_name>.yaml
```

Modifica il file YAML generato e aggiungi il campo `includedVirtualMachines` sotto `spec`, quindi applica il file:

```
kubectl apply -f <restore_name>.yaml
```

Esempio: Definisci un'applicazione basata su VM

La seguente applicazione protegge tre macchine virtuali specifiche in due namespace.

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: production-vms
  namespace: prod
spec:
  includedVirtualMachines:
    - namespace: prod
      names:
        - app-server
        - database-server
    - namespace: monitoring
      names:
```

```
        - prometheus-vm
  includedClusterScopedResources:
    - group: storage.k8s.io
      kind: StorageClass
```

Esempio: crea un backup

La creazione di un backup per un'applicazione basata su VM utilizza il CR di backup standard. Non sono richiesti campi specifici per la VM nel CR di backup.

```
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  name: daily-backup
  namespace: prod
spec:
  applicationRef: production-vms
  appVaultRef: s3-vault
```

Esempio: Ripristina una singola macchina virtuale da un backup

L'esempio seguente esegue il ripristino solo `database-server` dall'archivio di backup. Le altre VM nell'archivio, come `app-server` e `prometheus-vm`, vengono ignorate.

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: restore-db-only
  namespace: prod-vms-restore
spec:
  appVaultRef: s3-vault
  appArchivePath: backups/production-vms/2026-03-10T00-00-00Z
  namespaceMapping:
    - source: prod
      destination: prod-dr
  includedVirtualMachines:
    - namespace: prod
      names:
        - database-server
```

```
kubectl apply -f restore-db-only.yaml
```

Vengono ripristinati solo `database-server` e le sue risorse dipendenti, come DataVolumes, PVC, PV, Secrets, ConfigMaps e le risorse correlate di KubeVirt.

Esempio: Ripristina una singola macchina virtuale in-place da un backup

Il seguente esempio ripristina solo `database-server` nel suo spazio dei nomi originale dall'archivio di backup.

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: ipr-db-only
  namespace: prod
spec:
  appVaultRef: s3-vault
  appArchivePath: backups/production-vms/2026-03-10T00-00-00Z
  includedVirtualMachines:
    - namespace: prod
      names:
        - database-server
```

```
kubectl apply -f ipr-db-only.yaml
```

Esempio: Ripristina una singola macchina virtuale da uno Snapshot

L'esempio seguente esegue il ripristino solo `ubuntu-vm-blue-a627be38` da un archivio Snapshot.

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: blue-vm-snap-restore
  namespace: blue
spec:
  appVaultRef: my-appvault
  appArchivePath: snapshots/protectctl-blue-vm/snap-1
  includedVirtualMachines:
    - namespace: blue
      names:
        - ubuntu-vm-blue-a627be38
```

```
kubectl apply -f blue-vm-snap-restore.yaml
```

Combina il ripristino della macchina virtuale con altri filtri

Puoi usare `includedVirtualMachines` insieme a `resourceFilter`, `namespaceMapping` e `storageClassMapping` nello stesso restore CR per perfezionare cosa viene ripristinato.

Quando aggiungi un `resourceFilter`, Trident Protect seleziona prima le VM elencate in `includedVirtualMachines` e le relative risorse dipendenti. Poi applica il filtro delle risorse per restringere ulteriormente ciò che è incluso o escluso da quei risultati.

```
spec:
  includedVirtualMachines:
    - namespace: <vm_namespace>
      names:
        - <vm_name>
  resourceFilter:
    resourceSelectionCriteria: Include
    resourceMatchers:
      - kinds: ["PersistentVolumeClaim"]
        names: ["<pvc_name>"]
```

Per `BackupRestore`, `SnapshotRestore` e `ReplicateSnapshotRestore`, `namespaceMapping` e `storageClassMapping` lavorano insieme a `includedVirtualMachines`. I riferimenti al namespace in `includedVirtualMachines` corrispondono ai namespace di origine così come appaiono nell'archivio. Le mappature tra namespace e storage class vengono applicate durante la fase di trasformazione del ripristino dopo che il filtraggio è stato completato.

Per i dettagli sul comportamento di ripristino, consulta "Ripristina le applicazioni utilizzando Trident Protect".

Crea un'applicazione con ambito VM utilizzando la CLI

Usa il `--virtual-machines` flag per limitare l'ambito di un'applicazione a specifiche KubeVirt VM invece che a interi namespace. Il `--virtual-machines` flag è mutuamente esclusivo con `--namespaces`.

Fasi

1. Crea l'applicazione eseguendo uno dei seguenti comandi, sostituendo i valori tra parentesi con le informazioni del tuo ambiente. Il formato per il `--virtual-machines` flag è `<namespace>(<vm_name>)`. Per più VM, usa nomi separati da virgole all'interno delle parentesi:
   - Singola VM in un namespace:

     ```
     tridentctl-protect create application <application_name> \
       -n <application_namespace> \
       --virtual-machines "<vm_namespace>(<vm_name>)"
     ```

   - Più macchine virtuali in diversi namespace:

     ```
     tridentctl-protect create application <application_name> \
       -n <application_namespace> \
       --virtual-machines "<vm_namespace_1>(<vm_name_1>,<vm_name_2>
     ),<vm_namespace_2>(<vm_name_3>)"
     ```

Informazioni correlate

- "Definire un'applicazione per la gestione"
- "Proteggere le applicazioni"
- "Ripristina le applicazioni utilizzando Trident Protect"
- "Usa le impostazioni di ripristino avanzate"
- "Replicare le applicazioni utilizzando NetApp SnapMirror e Trident Protect"
- "Installa il plugin Trident Protect CLI"
- "Utilizzare gli oggetti AppVault per gestire i bucket"

Gestire gli hook di esecuzione Trident Protect

Un gancio di esecuzione è un'azione personalizzata che è possibile configurare per l'esecuzione in combinazione con un'operazione di protezione dei dati di un'applicazione gestita. Ad esempio, se si dispone di un'applicazione di database, è possibile utilizzare un gancio di esecuzione per mettere in pausa tutte le transazioni del database prima di uno snapshot e riprendere le transazioni al termine dello snapshot. Ciò garantisce snapshot coerenti con l'applicazione.

Tipi di hook di esecuzione

Trident Protect supporta i seguenti tipi di hook di esecuzione, in base al momento in cui possono essere eseguiti:

- Pre-snapshot
- Post-snapshot
- Pre-backup
- Post-backup
- Post-ripristino
- Post-failover

Ordine di esecuzione

Quando viene eseguita un'operazione di protezione dei dati, gli eventi hook di esecuzione hanno luogo nel seguente ordine:

1. Gli eventuali hook di esecuzione pre-operation personalizzati applicabili vengono eseguiti sui container appropriati. È possibile creare ed eseguire tutti gli hook pre-operation personalizzati necessari, ma l'ordine di esecuzione di questi hook prima dell'operazione non è garantito né configurabile.
2. Se applicabile, si verificano blocchi del file system. "Scopri di più sulla configurazione del congelamento del file system con Trident Protect".
3. Viene eseguita l'operazione di protezione dei dati.
4. I filesystem congelati vengono scongelati, se applicabile.
5. Gli eventuali hook di esecuzione post-operation personalizzati applicabili vengono eseguiti sui container appropriati. È possibile creare ed eseguire tutti gli hook post-operation personalizzati necessari, ma l'ordine di esecuzione di questi hook dopo l'operazione non è garantito né configurabile.

Se si creano più hook di esecuzione dello stesso tipo (ad esempio, pre-snapshot), l'ordine di esecuzione di tali hook non è garantito. Tuttavia, è garantito l'ordine di esecuzione di ganci di tipi diversi. Ad esempio, di seguito è riportato l'ordine di esecuzione di una configurazione che ha tutti i diversi tipi di ganci:

1. Hook pre-snapshot eseguiti
2. Esecuzione di hook post-snapshot
3. Hook pre-backup eseguiti
4. Hook post-backup eseguiti

NOTA	L'esempio dell'ordine precedente si applica solo quando si esegue un backup che non utilizza uno snapshot esistente.
NOTA	Prima di abilitarli in un ambiente di produzione, è necessario verificare sempre gli script hook di esecuzione. È possibile utilizzare il comando 'kubectl exec' per testare comodamente gli script. Dopo aver attivato gli hook di esecuzione in un ambiente di produzione, testare le snapshot e i backup risultanti per assicurarsi che siano coerenti. Per eseguire questa operazione, clonare l'applicazione in uno spazio dei nomi temporaneo, ripristinare lo snapshot o il backup e quindi testare l'applicazione.
NOTA	Se un gancio di esecuzione pre-snapshot aggiunge, modifica o rimuove le risorse Kubernetes, queste modifiche sono incluse nella snapshot o nel backup e in qualsiasi operazione di ripristino successiva.

Note importanti sugli hook di esecuzione personalizzati

Quando si pianificano gli hook di esecuzione per le applicazioni, considerare quanto segue.

- Un gancio di esecuzione deve utilizzare uno script per eseguire le azioni. Molti hook di esecuzione possono fare riferimento allo stesso script.
- Trident Protect richiede che gli script utilizzati dagli hook di esecuzione siano scritti nel formato degli script shell eseguibili.
- La dimensione dello script è limitata a 96 KB.
- Trident Protect utilizza le impostazioni dell'hook di esecuzione e tutti i criteri corrispondenti per determinare quali hook sono applicabili a un'operazione di snapshot, backup o ripristino.

NOTA	Poiché gli hook di esecuzione spesso riducono o disattivano completamente le funzionalità dell'applicazione con cui vengono eseguiti, si consiglia di ridurre al minimo il tempo necessario per l'esecuzione degli hook di esecuzione personalizzati. Se si avvia un'operazione di backup o snapshot con gli hook di esecuzione associati, ma poi si annulla, gli hook possono ancora essere eseguiti se l'operazione di backup o snapshot è già iniziata. Ciò significa che la logica utilizzata in un gancio di esecuzione post-backup non può presumere che il backup sia stato completato.

Esecuzione dei filtri hook

Quando si aggiunge o si modifica un gancio di esecuzione per un'applicazione, è possibile aggiungere filtri al gancio di esecuzione per gestire i contenitori corrispondenti. I filtri sono utili per le applicazioni che utilizzano la stessa immagine container su tutti i container, ma possono utilizzare ogni immagine per uno scopo diverso (ad esempio Elasticsearch). I filtri consentono di creare scenari in cui gli hook di esecuzione vengono eseguiti su

alcuni container identici, ma non necessariamente su tutti. Se si creano più filtri per un singolo gancio di esecuzione, questi vengono combinati con un operatore AND logico. È possibile avere fino a 10 filtri attivi per gancio di esecuzione.

Ogni filtro aggiunto a un hook di esecuzione utilizza un'espressione regolare per abbinare i contenitori nel cluster. Quando un hook corrisponde a un contenitore, eseguirà lo script associato su quel contenitore. Le espressioni regolari per i filtri utilizzano la sintassi Regular Expression 2 (RE2), che non supporta la creazione di un filtro che escluda i contenitori dall'elenco delle corrispondenze. Per informazioni sulla sintassi supportata da Trident Protect per le espressioni regolari nei filtri di hook di esecuzione, vedere "Supporto della sintassi RE2 (Regular Expression 2)" .

NOTA	Se si aggiunge un filtro dello spazio dei nomi a un gancio di esecuzione che viene eseguito dopo un'operazione di ripristino o clonazione e l'origine e la destinazione del ripristino o del clone si trovano in spazi dei nomi diversi, il filtro dello spazio dei nomi viene applicato solo allo spazio dei nomi di destinazione.

Esempi di gancio di esecuzione

Visita il sito "Progetto NetApp Verda GitHub" per scaricare i veri hook di esecuzione per le app più diffuse, come Apache Cassandra ed Elasticsearch. Puoi anche vedere esempi e trovare idee per strutturare i tuoi hook di esecuzione personalizzati.

NOTA	Gli script presenti nel repository Verda vengono forniti così come sono e non sono ufficialmente supportati da NetApp. Il supporto NetApp copre solo il framework degli hook di esecuzione all'interno di Trident Protect.

Creare un gancio di esecuzione

È possibile creare un hook di esecuzione personalizzato per un'app utilizzando Trident Protect. Per creare hook di esecuzione è necessario disporre delle autorizzazioni di Proprietario, Amministratore o Membro.

Utilizzare un CR

Fasi

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-hook.yaml`.
2. Configurare i seguenti attributi in modo che corrispondano all'ambiente Trident Protect e alla configurazione del cluster:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.applicationRef**: (*required*) il nome Kubernetes dell'applicazione per la quale eseguire l'hook di esecuzione.
   - **Spec.stage**: (*required*) stringa che indica quale fase durante l'azione deve essere eseguita l'hook di esecuzione. Valori possibili:
     - Pre
     - Post
   - **Spec.action**: (*required*) stringa che indica l'azione che verrà eseguita dall'hook di esecuzione, presupponendo che tutti i filtri di hook di esecuzione specificati siano corrispondenti. Valori possibili:

        - Snapshot
        - Backup
        - Ripristinare
        - Failover
    - **Spec.Enabled**: (*Optional*) indica se questo gancio di esecuzione è abilitato o disabilitato. Se non specificato, il valore predefinito è true.
    - **Spec.hookSource**: (*required*) stringa contenente lo script hook codificato in base64.
    - **Spec.timeout**: (*Optional*) Un numero che definisce il tempo in minuti per il quale il gancio di esecuzione può essere eseguito. Il valore minimo è 1 minuto e, se non specificato, il valore predefinito è 25 minuti.
    - **Spec.arguments**: (*Optional*) elenco YAML di argomenti che è possibile specificare per l'hook di esecuzione.
    - **Spec.matchingCriteria**: (*Optional*) un elenco facoltativo di coppie di valori chiave di criteri, ciascuna coppia costituendo un filtro di hook di esecuzione. È possibile aggiungere fino a 10 filtri per ogni collegamento di esecuzione.
    - **Spec.matchingCriteria.type**: (*Optional*) Una stringa che identifica il tipo di filtro del gancio di esecuzione. Valori possibili:
        - Immagine containerImage
        - ContainerName
        - PodName
        - PodLabel
        - NamespaceName
    - **Spec.matchingCriteria.value**: (*Optional*) Una stringa o Un'espressione regolare che identifica il valore del filtro dell'hook di esecuzione.

    Esempio YAML:

```
apiVersion: protect.trident.netapp.io/v1
kind: ExecHook
metadata:
  name: example-hook-cr
  namespace: my-app-namespace
  annotations:
    astra.netapp.io/astra-control-hook-source-id:
/account/test/hookSource/id
spec:
  applicationRef: my-app-name
  stage: Pre
  action: Snapshot
  enabled: true
  hookSource: IyEvYmluL2Jhc2gKZWNobyAiZXhhbXBsZSBzY3JpcHQiCg==
  timeout: 10
  arguments:
    - FirstExampleArg
```

```
    - SecondExampleArg
  matchingCriteria:
    - type: containerName
      value: mysql
    - type: containerImage
      value: bitnami/mysql
    - type: podName
      value: mysql
    - type: namespaceName
      value: mysql-a
    - type: podLabel
      value: app.kubernetes.io/component=primary
    - type: podLabel
      value: helm.sh/chart=mysql-10.1.0
    - type: podLabel
      value: deployment-type=production
```

3. Dopo aver popolato il file CR con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-hook.yaml
```

Utilizzare la CLI

Fasi

1. Creare il gancio di esecuzione, sostituendo i valori tra parentesi con le informazioni dell'ambiente. Ad esempio:

```
tridentctl-protect create exechook <my_exec_hook_name> --action
<action_type> --app <app_to_use_hook> --stage <pre_or_post_stage>
--source-file <script-file> -n <application_namespace>
```

Eseguire manualmente un gancio di esecuzione

È possibile eseguire manualmente un gancio di esecuzione a scopo di test o se è necessario eseguire nuovamente il gancio manualmente dopo un errore. È necessario disporre delle autorizzazioni Proprietario, Amministratore o membro per eseguire manualmente i ganci di esecuzione.

L'esecuzione manuale di un gancio di esecuzione consiste in due passaggi di base:

1. Creare un backup delle risorse, che raccoglie le risorse e ne crea un backup, determinando dove verrà eseguito il hook
2. Eseguire il gancio di esecuzione sul backup

▼ Passaggio 1: Creare un backup delle risorse

Utilizzare un CR

Fasi

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-resource-backup.yaml`.
2. Configurare i seguenti attributi in modo che corrispondano all'ambiente Trident Protect e alla configurazione del cluster:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.applicationRef**: (*required*) il nome Kubernetes dell'applicazione per cui creare il backup delle risorse.
   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui sono memorizzati i contenuti di backup.
   - **Spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono memorizzati i contenuti di backup. Per trovare il percorso, utilizzare il seguente comando:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

     Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ResourceBackup
metadata:
  name: example-resource-backup
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
```

3. Dopo aver popolato il file CR con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-resource-backup.yaml
```

Utilizzare la CLI

Fasi

1. Creare il backup, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente. Ad esempio:

```
tridentctl protect create resourcebackup <my_backup_name> --app <my_app_name> --appvault <my_appvault_name> -n <my_app_namespace>
```

```
--app-archive-path <app_archive_path>
```

2. Visualizzare lo stato del backup. È possibile utilizzare questo comando di esempio ripetutamente fino al completamento dell'operazione:

```
tridentctl protect get resourcebackup -n <my_app_namespace>
<my_backup_name>
```

3. Verificare che il backup sia stato eseguito correttamente:

```
kubectl describe resourcebackup <my_backup_name>
```

▼ **Fase 2: Eseguire il gancio di esecuzione**

Utilizzare un CR

Fasi

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-hook-run.yaml`.
2. Configurare i seguenti attributi in modo che corrispondano all'ambiente Trident Protect e alla configurazione del cluster:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.applicationRef**: (*required*) assicurarsi che questo valore corrisponda al nome dell'applicazione dal ResourceBackup CR creato nel passaggio 1.
   - **Spec.appVaultRef**: (*required*) assicurarsi che questo valore corrisponda all'appVaultRef del ResourceBackup CR creato nel passaggio 1.
   - **Spec.appArchivePath**: Assicurarsi che questo valore corrisponda all'appArchivePath del ResourceBackup CR creato nel passaggio 1.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.action**: (*required*) stringa che indica l'azione che verrà eseguita dall'hook di esecuzione, presupponendo che tutti i filtri di hook di esecuzione specificati siano corrispondenti. Valori possibili:
     - Snapshot
     - Backup
     - Ripristinare
     - Failover
   - **Spec.stage**: (*required*) stringa che indica quale fase durante l'azione deve essere eseguita l'hook di esecuzione. Questa corsa del gancio non farà funzionare i ganci in nessun altro

stadio. Valori possibili:

- Pre
- Post

Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ExecHooksRun
metadata:
  name: example-hook-run
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
  stage: Post
  action: Failover
```

3. Dopo aver popolato il file CR con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-hook-run.yaml
```

Utilizzare la CLI

Fasi

1. Creare la richiesta di esecuzione hook manuale:

```
tridentctl protect create exechooksrun <my_exec_hook_run_name> -n
<my_app_namespace> --action snapshot --stage <pre_or_post> --app
<my_app_name> --appvault <my_appvault_name> --path
<my_backup_name>
```

2. Controllare lo stato della sequenza di aggancio esecuzione. È possibile eseguire questo comando ripetutamente fino al completamento dell'operazione:

```
tridentctl protect get exechooksrun -n <my_app_namespace>
<my_exec_hook_run_name>
```

3. Descrivere l'oggetto exechooksrun per visualizzare i dettagli e lo stato finali:

```
kubectl -n <my_app_namespace> describe exechooksrun
<my_exec_hook_run_name>
```

Disinstallare Trident Protect

Potrebbe essere necessario rimuovere i componenti Trident Protect se si esegue l'aggiornamento da una versione di prova a una versione completa del prodotto.

Per rimuovere Trident Protect, procedere come segue.

Fasi

1. Rimuovere i file CR Trident Protect:

   NOTA Questo passaggio non è necessario per la versione 25.06 e successive.

   ```
   helm uninstall -n trident-protect trident-protect-crds
   ```

2. Rimuovere Trident Protect:

   ```
   helm uninstall -n trident-protect trident-protect
   ```

3. Rimuovere lo spazio dei nomi Trident Protect:

   ```
   kubectl delete ns trident-protect
   ```

Blog Trident e Trident Protect

Puoi trovare alcuni ottimi blog NetApp Trident e Trident Protect qui:

Blog Trident

- 16 ottobre 2025:"Soluzioni di archiviazione avanzate per Kubernetes"
- 19 agosto 2025:"Miglioramento della coerenza dei dati: snapshot del gruppo di volumi nella virtualizzazione OpenShift con Trident"
- 9 maggio 2025: "Configurazione automatica del backend Trident per FSx per ONTAP con il componente aggiuntivo Amazon EKS"
- 15 aprile 2025: "NetApp Trident con Google Cloud NetApp Volumes per il protocollo SMB"
- 14 aprile 2025: "Utilizzo del protocollo Fibre Channel con Trident 25.02 per l'archiviazione persistente su Kubernetes"
- 14 aprile 2025: "Sfruttare la potenza dei sistemi NetApp ASA r2 per l'archiviazione a blocchi di Kubernetes"
- 31 marzo 2025: "Semplificazione dell'installazione Trident su Red Hat OpenShift con il nuovo operatore certificato"
- 27 marzo 2025: "Provisioning Trident per PMI con Google Cloud NetApp Volumes"
- 5 marzo 2025: "Sblocca la perfetta integrazione dello storage iSCSI: Guida a FSxN su cluster ROSA per AWS"
- 27 febbraio 2025: "Distribuzione dell'identità cloud con Trident, GKE e Google Cloud NetApp Volumes"
- 12 dicembre 2024: "Presentazione del supporto Fibre Channel in Trident"
- 11 novembre 2024: "NetApp Trident con Google Cloud NetApp Volumes"
- 29 ottobre 2024: "Amazon FSx for NetApp ONTAP con Red Hat OpenShift Service su AWS (ROSA) utilizzando Trident"
- 29 ottobre 2024: "Migrazione live di VM con OpenShift Virtualization su ROSA e Amazon FSx for NetApp ONTAP"
- 8 luglio 2024: "Utilizzo di NVMe/TCP per consumare lo storage ONTAP per le tue moderne app containerizzate su Amazon EKS"
- 1 luglio 2024: "Archiviazione Kubernetes senza interruzioni con Google Cloud NetApp Volumes Flex e Astra Trident"
- 11 giugno 2024: "ONTAP come archivio backend per il registro immagini integrato in OpenShift"

Blog Trident Protect

- 16 maggio 2025:"Automazione del failover del registro per il ripristino di emergenza con hook post-ripristino Trident Protect"
- 16 maggio 2025: "Ripristino di emergenza della virtualizzazione OpenShift con NetApp Trident Protect"
- 13 maggio 2025:"Migrazione della classe di archiviazione con backup e ripristino Trident Protect"
- 9 maggio 2025:"Ridimensiona le applicazioni Kubernetes con gli hook post-ripristino Trident Protect"
- 3 aprile 2025: "Trident Protect Power Up: replica di Kubernetes per protezione e ripristino di emergenza"
- 13 marzo 2025: "Operazioni di backup e ripristino coerenti con il crash per le VM di virtualizzazione

OpenShift"

- 11 marzo 2025: "Estensione dei modelli GitOps alla data Protection applicativa con NetApp Trident"
- 03 marzo 2025: "Trident 25,02: Migliorare l'esperienza Red Hat OpenShift con nuove ed entusiasmanti funzionalità"
- 15 gennaio 2025:"Presentazione del controllo degli accessi basato sui ruoli Trident Protect"
- 11 novembre 2024:"Gestione dei dati basata su Kubernetes: la nuova era con Trident Protect"

Conoscenza e supporto

Domande frequenti

Trova le risposte alle domande frequenti sull'installazione, la configurazione, l'aggiornamento e la risoluzione dei problemi di Trident.

Domande generali

Con quale frequenza viene rilasciato Trident?

A partire dalla versione 24,02, Trident viene rilasciato ogni quattro mesi: Febbraio, giugno e ottobre.

Trident supporta tutte le funzionalità rilasciate in una particolare versione di Kubernetes?

In genere, Trident non supporta le funzionalità alfa in Kubernetes. Trident potrebbe supportare le funzionalità beta all'interno delle due release Trident che seguono la release beta di Kubernetes.

Trident dipende da altri prodotti NetApp per il suo funzionamento?

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 si ottengono i dettagli completi della configurazione di Trident?

Utilizzare il `tridentctl get` comando per ottenere ulteriori informazioni sulla configurazione di Trident.

Posso ottenere delle metriche sul provisioning dello storage da parte di Trident?

Sì. Endpoint Prometheus che possono essere utilizzati per raccogliere informazioni sul funzionamento di Trident, come il numero di backend gestiti, il numero di volumi sottoposti a provisioning, i byte consumati e così via. È inoltre possibile utilizzare "Cloud Insights" per il monitoraggio e l'analisi.

L'esperienza utente cambia quando si utilizza Trident come revisioner CSI?

No. Non ci sono modifiche per quanto riguarda l'esperienza utente e le funzionalità. Il nome del fornitore utilizzato è `csi.trident.netapp.io`. Questo metodo di installazione di Trident è consigliato se si desidera utilizzare tutte le nuove funzioni fornite dalle versioni attuali e future.

Installare e utilizzare Trident su un cluster Kubernetes

Trident supporta un'installazione offline da un registro privato?

Sì, Trident può essere installato offline. Consulta "Informazioni sull'installazione di Trident".

È possibile installare Trident BE in remoto?

Sì. Trident 18,10 e versioni successive supportano funzionalità di installazione remota da qualsiasi computer con `kubectl` accesso al cluster. Dopo aver `kubectl` verificato l'accesso (ad esempio, avviare un `kubectl get nodes` comando dal computer remoto per verificare), seguire le istruzioni di installazione.

Posso configurare alta disponibilità con Trident?

Trident viene installato come distribuzione Kubernetes (ReplicaSet) con un'istanza e dispone di ha incorporato. Non è necessario aumentare il numero di repliche nella distribuzione. Se il nodo in cui è installato Trident viene perso o il pod è altrimenti inaccessibile, Kubernetes ridistribuisce automaticamente il pod in un nodo integro nel cluster. Trident è solo per il piano di controllo, pertanto i pod attualmente montati non sono interessati se Trident viene riimplementato.

Trident ha bisogno di accedere allo spazio dei nomi kube-System?

Trident legge dal server API Kubernetes per determinare quando le applicazioni richiedono nuovi PVC, in modo che abbia bisogno di accedere al kube-system.

Quali sono i ruoli e Privileges utilizzati da Trident?

Il programma di installazione di 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 manifesti esatti utilizzati da Trident per l'installazione?

È possibile generare e modificare localmente i file manifest esatti utilizzati da Trident per l'installazione, se necessario. Fare riferimento a "Personalizzare l'installazione di tridentctl".

Posso condividere la stessa SVM di back-end ONTAP per due istanze Trident separate per due cluster Kubernetes separati?

Anche se non è consigliabile, puoi utilizzare la stessa SVM di back-end per due istanze Trident. Specificare un nome di volume univoco per ogni istanza durante l'installazione e/o specificare un parametro univoco `StoragePrefix` nel `setup/backend.json` file. In questo modo si garantisce che lo stesso FlexVol volume non venga utilizzato per entrambe le istanze.

È possibile installare Trident in ContainerLinux (in precedenza CoreOS)?

Trident è semplicemente un pod Kubernetes e può essere installato ovunque venga eseguito Kubernetes.

È possibile utilizzare Trident con NetApp Cloud Volumes ONTAP?

Sì, Trident è supportato su AWS, Google Cloud e Azure.

Risoluzione dei problemi e supporto

NetApp supporta Trident?

Anche se Trident è open source e fornito gratuitamente, NetApp lo supporta completamente a condizione che il backend 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 "Trident Github" e citare **RFE** nell'oggetto e nella descrizione del problema.

Dove posso segnalare un difetto?

Creare un problema su "Trident Github". Assicurarsi di includere tutte le informazioni e i registri necessari relativi al problema.

Cosa succede se ho una domanda rapida su Trident su cui ho bisogno di chiarimenti? Esiste una community o un forum?

In caso di domande, problemi o richieste, contattaci tramite il nostro Trident "Discordare il canale"o GitHub.

La password del mio sistema storage è cambiata e Trident non funziona più. Come posso ripristinarla?

Aggiornare la password del backend con `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`. Sostituire `myBackend` nell'esempio con il nome backend, e. `/path/to_new_backend.json` con il percorso verso il corretto `backend.json` file.

Trident non riesce a trovare il nodo Kubernetes. Come posso risolvere questo problema?

Esistono due scenari probabili per cui 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 sono verificate modifiche alla rete dopo l'installazione di Trident, si riscontra questo problema 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.

Upgrade Trident (Aggiorna server)

È possibile eseguire l'aggiornamento da una versione precedente direttamente a una versione più recente (ignorando alcune versioni)?

NetApp supporta l'aggiornamento di Trident da una release principale alla release principale successiva. 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 Trident"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 sia la gestione che i dati in un file di definizione back-end ONTAP?

La LIF di gestione è obbligatoria. La DataLIF varia:

- ONTAP SAN (SAN iSCSI): Non specificare iSCSI. Trident utilizza "Mappa LUN selettiva ONTAP" per scoprire le interfacce LIF isci necessarie per stabilire una sessione multipercorso. Viene generato un avviso se `dataLIF` è definito esplicitamente. Per ulteriori informazioni, fare riferimento alla "Opzioni ed esempi di configurazione DELLA SAN ONTAP" sezione.
- ONTAP NAS: NetApp consiglia di specificare `dataLIF`. Se non viene fornita, Trident recupera le LIF dati dalla SVM. È possibile specificare un nome di dominio completo (FQDN) da utilizzare per le operazioni di montaggio NFS, consentendo di creare un DNS round-robin per bilanciare il carico su più LIF dati. Per ulteriori informazioni, fare riferimento alla sezione"Opzioni ed esempi di configurazione del NAS ONTAP"

Trident può configurare CHAP per i backend ONTAP?

Sì. Trident supporta il protocollo CHAP bidirezionale per i backend ONTAP. Questa operazione richiede l'impostazione `useCHAP=true` nella configurazione backend.

Come posso gestire i criteri di esportazione con Trident?

Trident è in grado di creare e gestire dinamicamente i criteri di esportazione dalla versione 20,04 in poi. 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, Trident gestisce automaticamente l'aggiunta e l'eliminazione di regole per i nodi con IP all'interno dei CIDR specificati.

È possibile utilizzare gli indirizzi IPv6 per la gestione e DataLIF?

Trident supporta la definizione di indirizzi IPv6 per:

- `managementLIF` e. `dataLIF` Per backend NAS ONTAP.
- `managementLIF` Per backend SAN ONTAP. Impossibile specificare `dataLIF` Su un backend SAN ONTAP.

Trident deve essere installato utilizzando il flag `--use-ipv6` (per `tridentctl` l'installazione), `IPv6` (per l'operatore Trident) o `tridentTPv6` (per l'installazione di Helm) perché funzioni su IPv6.

È possibile aggiornare la LIF di gestione sul back-end?

Sì, è possibile aggiornare la LIF di gestione back-end utilizzando `tridentctl update backend` comando.

È possibile aggiornare DataLIF nel back-end?

È possibile aggiornare il DataLIF solo su `ontap-nas` e. `ontap-nas-economy`

Posso creare diversi backend in Trident per Kubernetes?

Trident può supportare più backend contemporaneamente, con lo stesso driver o driver diversi.

In che modo Trident archivia le credenziali backend?

Trident memorizza le credenziali backend come Kubernetes Secrets.

In che modo 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 Trident non esegua il provisioning da un backend specifico?

Il `excludeStoragePools` parametro viene utilizzato per filtrare l'insieme di pool utilizzato da Trident per il provisioning e rimuoverà tutti i pool corrispondenti.

Se esistono più backend dello stesso tipo, in che modo Trident seleziona quale backend utilizzare?

Se sono presenti più backend configurati dello stesso tipo, 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, Trident tenta di far corrispondere i parametri in `StorageClass` e combinati e `PersistentVolumeClaim` di far corrispondere un backend in grado di soddisfare i requisiti elencati in `StorageClass` e `PersistentVolumeClaim`. Se sono presenti più backend che corrispondono alla richiesta, Trident seleziona uno di essi in modo casuale.

Trident supporta CHAP bidirezionale con Element/SolidFire?

Sì.

In che modo Trident implementa Qtree su un volume ONTAP? Quanti Qtree possono essere implementati su un singolo volume?

```
`ontap-nas-economy`Il driver crea fino a 200 Qtree nella stessa FlexVol
volume (configurabile tra 50 e 300), 100.000 Qtree per nodo del cluster e
2,4M TB per cluster. Quando si immette un nuovo `PersistentVolumeClaim`
che viene gestito dal driver Economy, il conducente cerca di vedere se
esiste già un FlexVol volume in grado di servire il nuovo Qtree. Se il
FlexVol volume non esiste e può servire la Qtree, viene creato un nuovo
FlexVol volume.
```

Come si impostano le autorizzazioni Unix per i volumi forniti su NAS ONTAP?

È possibile impostare autorizzazioni Unix sul volume fornito da Trident impostando un parametro nel file di definizione backend.

Come posso configurare un set esplicito di opzioni di montaggio NFS di ONTAP durante il provisioning di un volume?

Per impostazione predefinita, Trident non imposta le opzioni di montaggio su alcun valore con Kubernetes. Per specificare le opzioni di montaggio nella classe di archiviazione 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 dei volumi tramite 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, consultare: "Come funziona Trident con NVE e NAE"

Qual è il modo migliore per implementare QoS per ONTAP tramite Trident?

Utilizzare `StorageClasses` Per implementare QoS per ONTAP.

Come è possibile specificare il thin provisioning o il thick provisioning tramite 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.

È possibile far crescere il numero di PVC NFS creati da Trident?

Sì. Puoi espandere un PVC NFS creato da Trident. Le policy di crescita automatica di Trident possono espandere automaticamente i PVC in base alle soglie di capacità. Il dimensionamento automatico del volume ONTAP è separato e si applica ai driver SAN ONTAP supportati: aumenta il FlexVol di backing senza modificare le dimensioni del PVC. Per ulteriori informazioni, vedi "ONTAP dimensionamento automatico".

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 è in grado di rispettare le impostazioni della quota Kubernetes a causa della mancanza di capacità, Trident tenta di eseguire il provisioning, con errori che vengono eliminati.

È possibile creare snapshot del volume utilizzando Trident?

Sì. La creazione di snapshot di volumi on-demand e di volumi persistenti da Snapshot sono supportate da Trident. Per creare PVS dalle istantanee, assicurarsi che il `VolumeSnapshotDataSource` gate delle funzioni sia stato attivato.

Quali sono i driver che supportano le snapshot di volume Trident?

Da oggi, il supporto snapshot on-demand è disponibile per il nostro `ontap-nas` , `ontap-nas-flexgroup` , `ontap-san` , `ontap-san-economy` , `solidfire-san` , E `azure-netapp-files` driver backend.

Come è possibile eseguire un backup snapshot di un volume dotato di provisioning 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 operazione è disponibile anche `ontap-nas-economy` sui driver, ma non sulla granularità a livello di FlexVol volume e non a livello di qtree. Per abilitare la possibilità di creare snapshot dei volumi forniti da Trident, imposta l'opzione del parametro backend `snapshotPolicy` sulla policy dello snapshot desiderata, come definito nel back-end ONTAP. Trident non conosce istantanee scattate dallo storage controller.

È possibile impostare una percentuale di riserva di snapshot per un volume sottoposto a provisioning tramite Trident?

Sì, è possibile riservare una percentuale specifica di spazio su disco per l'archiviazione delle copie snapshot tramite Trident impostando l' `snapshotReserve`attributo nel file di definizione backend. Se è stato configurato `snapshotPolicy` e `snapshotReserve` nel file di definizione backend, la percentuale di riserva snapshot viene impostata in base alla `snapshotReserve` percentuale indicata nel file backend. Se il `snapshotReserve` numero di percentuale non viene menzionato, ONTAP utilizza per impostazione predefinita la percentuale di riserva dello snapshot come 5. Se l' `snapshotPolicy`opzione è impostata su nessuno, la percentuale di riserva istantanea 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.

Posso configurare SnapMirror per i volumi tramite 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

Per la risoluzione dei problemi che si possono verificare durante l'installazione e l'utilizzo di Trident, utilizzare i puntatori forniti di seguito.

NOTA Per ottenere assistenza con Trident, creare un bundle di supporto utilizzando `tridentctl logs -a -n trident` e inviarlo al supporto NetApp.

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 Trident 20,07 e 20,10 utilizzare `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 registri di debug per ogni backend includendo `debugTraceFlags` nella definizione di backend. Ad esempio, includere `debugTraceFlags: {"api":true, "method":true,}` per ottenere le chiamate API e i percorsi del metodo nei registri Trident. I backend esistenti possono essere `debugTraceFlags` configurati con un `tridentctl backend update`.
- Quando si utilizza Red Hat Enterprise Linux CoreOS (RHCOS), assicurarsi che `iscsid` sia abilitato sui nodi di lavoro e avviato per impostazione predefinita. Questa operazione può essere eseguita utilizzando OpenShift MachineConfigs o modificando i modelli di accensione.
- Un problema comune che potresti riscontrare quando utilizzi Trident "Azure NetApp Files" è quando i tenant e i segreti del client provengono da una registrazione dell'app con autorizzazioni insufficienti. Per un elenco completo dei requisiti di Trident, consulta la "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, consultare `TridentOrchestrator` stato.

```
kubectl describe torc trident-2
Name:         trident-2
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Status:
  Current Installation Params:
    IPv6:
    Autosupport Hostname:
    Autosupport Image:
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:
    Image Pull Secrets:         <nil>
    Image Registry:
    k8sTimeout:
    Kubelet Dir:
    Log Format:
    Silence Autosupport:
    Trident Image:
  Message:                      Trident is bound to another CR 'trident'
  Namespace:                    trident-2
  Status:                       Error
  Version:
Events:
  Type     Reason  Age                From                           Message
  ----     ------  ----               ----                           -------
  Warning  Error   16s (x2 over 16s)  trident-operator.netapp.io  Trident
is bound to another CR 'trident'
```

Questo errore indica che esiste già un `TridentOrchestrator`Utilizzato per installare Trident. Poiché ogni cluster Kubernetes può avere una sola istanza di Trident, l'operatore garantisce che in qualsiasi momento esista una sola istanza attiva `TridentOrchestrator` che può creare.

Inoltre, osservare lo stato dei pod Trident può spesso indicare se qualcosa non è giusto.

```
kubectl get pods -n trident

NAME                                READY   STATUS             RESTARTS
AGE
trident-csi-4p5kq                   1/2     ImagePullBackOff   0
5m18s
trident-csi-6f45bfd8b6-vfrkw        4/5     ImagePullBackOff   0
5m19s
trident-csi-9q5xc                   1/2     ImagePullBackOff   0
5m18s
trident-csi-9v95z                   1/2     ImagePullBackOff   0
5m18s
trident-operator-766f7b8658-ldzsv   1/1     Running            0
8m17s
```

È possibile notare che i pod non sono in grado di inizializzare completamente perché una o più immagini container non sono state recuperate.

Per risolvere il problema, modificare `TridentOrchestrator` CR. In alternativa, è possibile eliminare `TridentOrchestrator`e crearne uno nuovo con la definizione modificata e precisa.

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 Trident e CRD

È possibile rimuovere completamente Trident e tutti i CRD creati e le risorse personalizzate associate.

ATTENZIONE	Questa operazione non può essere annullata. Non eseguire questa operazione a meno che non si desideri una nuova installazione di Trident. Per disinstallare Trident senza rimuovere i CRD, fare riferimento a "Disinstallare Trident".

Operatore Trident

Per disinstallare 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 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 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 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 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.
```

I client NFSv4.2 segnalano "argomento non valido" dopo l'aggiornamento ONTAP quando si prevede che "v4.2-xattrs" sia abilitato

Dopo l'aggiornamento ONTAP, i client NFSv4.2 potrebbero segnalare errori di tipo "argomento non valido" quando tentano di montare esportazioni NFSv4.2. Questo problema si verifica quando il `v4.2-xattrs` l'opzione non è abilitata sulla SVM. .Soluzione alternativa Abilitare il `v4.2-xattrs` opzione sull'SVM o eseguire l'aggiornamento a ONTAP 9.12.1 o versione successiva, dove questa opzione è abilitata per impostazione predefinita.

Supporto

NetApp offre supporto per Trident in diversi 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 Trident

Trident offre tre livelli di supporto in base alla tua versione. Fare riferimento alla "Supporto delle versioni software NetApp per le definizioni".

Supporto completo
: Trident fornisce supporto completo per dodici mesi dalla data di rilascio.

Supporto limitato
: Trident fornisce supporto limitato per i mesi dal 13 al 24 dalla data di rilascio.

Supporto autonomo
: La documentazione Trident è disponibile per i mesi dal 25 al 36 dalla data di rilascio.

Versione	Fine del supporto completo	Fine del supporto limitato	Fine del supporto autonomo
"26,10"	Ottobre 2027	Ottobre 2028	Ottobre 2029
"26,06"	Luglio 2027	Luglio 2028	Luglio 2029

"26,02"	Febbraio 2027	Febbraio 2028	Febbraio 2029
"25,10"	Ottobre 2026	Ottobre 2027	Ottobre 2028
"25,06"	Giugno 2026	Giugno 2027	Giugno 2028
"25,02"	Febbraio 2026	Febbraio 2027	Febbraio 2028
"24,10"	—	Ottobre 2026	Ottobre 2027
"24,06"	—	Giugno 2026	Giugno 2027
"24,02"	—	Febbraio 2026	Febbraio 2027
"23,10"	—	—	Ottobre 2026
"23,07"	—	—	Luglio 2026
"23.04"	—	—	Aprile 2026

Supporto autonomo

Per un elenco completo di articoli sulla risoluzione dei problemi, fare riferimento a "Knowledge base di NetApp (accesso richiesto)".

Sostegno della community

Esiste una vivace comunità pubblica di utenti di container (compresi gli sviluppatori Trident) sul nostro "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 Trident, creare un pacchetto di supporto utilizzando `tridentctl logs -a -n trident` e inviarlo a `NetApp Support <Getting Help>`.

Per ulteriori informazioni

- "Risorse Trident"
- "Hub Kubernetes"

Riferimento

Porte Trident

Ulteriori informazioni sulle porte utilizzate da Trident per la comunicazione.

Panoramica

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

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

Porte per i nodi controller

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

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

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

Porte per i nodi worker

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

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

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

NOTA 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 Trident

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

Quando utilizzare l'API REST

Le API REST sono utili per installazioni avanzate che utilizzano Trident come binario standalone in implementazioni non-Kubernetes.

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

Utilizzo dell'API REST

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

> **NOTA** 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 Trident usando tridentctl".

Opzioni della riga di comando

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>

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

Docker

-volume_driver <name>

Nome del driver utilizzato durante la registrazione del plugin Docker. 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.

AVVERTENZA	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`.

SUGGERIMENTO	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, 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

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
trident.netapp.io/skipRecoveryQueue	saltaCoda di Recupero	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy

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:

- NetApp consiglia di eseguire il cloning di 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.

NOTA	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
StoragePools	map[string]StringList	no	Mappatura dei nomi backend agli elenchi di pool di storage all'interno di
AddtionalStoragePools	map[string]StringList	no	Mappatura dei nomi backend agli elenchi di pool di storage all'interno di
EsclusiveStoragePools	map[string]StringList	no	Mappatura dei nomi backend agli elenchi di pool di storage all'interno di

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

Attributi di selezione del pool di storage

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

Attributo	Tipo	Valori	Offerta	Richiesta	Supportato da
supporti[1]	stringa	hdd, ibrido, ssd	Il pool contiene supporti di questo tipo; ibridi significa entrambi	Tipo di supporto specificato	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
ProvisioningType	stringa	sottile, spesso	Il pool supporta questo metodo di provisioning	Metodo di provisioning specificato	thick: all ONTAP; thin: all ONTAP e solidfire-san
BackendType	stringa	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, 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

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
crittografia	bool	vero, falso	Il pool supporta volumi crittografati	Volume con crittografia attivata	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	int	intero positivo	Il pool è in grado di garantire IOPS in questa gamma	Volume garantito per questi IOPS	solidfire-san

[1]: Non supportato dai sistemi ONTAP Select

Nella maggior parte dei casi, i valori richiesti influiscono direttamente sul provisioning; ad esempio, la richiesta di thick provisioning comporta un volume con provisioning spesso. Tuttavia, un pool di storage di elementi utilizza i valori IOPS minimi e massimi offerti per impostare i valori QoS, piuttosto che il valore richiesto. In questo caso, il valore richiesto viene utilizzato solo per selezionare il pool di storage.

Idealmente, è possibile utilizzare `attributes` da soli per modellare le qualità dello storage necessarie per soddisfare le esigenze di una particolare classe. Trident rileva e seleziona automaticamente i pool di storage che corrispondono a *tutti* di `attributes` specificato dall'utente.

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

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

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

È possibile utilizzare `excludeStoragePools` Parametro per filtrare il set di pool che Trident utilizza per il provisioning. L'utilizzo di questo parametro consente di rimuovere i pool corrispondenti.

In `storagePools` e. `additionalStoragePools` parametri, ogni voce assume la forma `<backend>:<storagePoolList>`, dove `<storagePoolList>` è un elenco separato da virgole di pool di storage per il backend specificato. Ad esempio, un valore per `additionalStoragePools` potrebbe sembrare `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`. Questi elenchi accettano valori regex sia per i valori di backend che per quelli di elenco. È possibile utilizzare `tridentctl get backend` per ottenere l'elenco dei backend e dei relativi pool.

Attributi Kubernetes

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

potrebbero richiedere utility del file system, come xfsprogs.

Attributo	Tipo	Valori	Descrizione	Driver pertinenti	Versione di Kubernetes
Fstype	stringa	ext4, ext3, xfs	Il tipo di file system per i volumi a blocchi	solidfire-san, ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy	Tutto
AllowVolumeExpansion	booleano	vero, falso	Abilitare o disabilitare il supporto per aumentare le dimensioni del PVC	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, 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

SUGGERIMENTO

- 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.

NOTA Il ciclo di vita di un VolumeSnapshot è indipendente dal PVC di origine: uno snapshot persiste anche dopo l'eliminazione 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.

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

Oggetti Kubernetes `VolumeGroupSnapshotClass`

Gli oggetti Kubernetes `VolumeGroupSnapshotClass` sono analoghi a `VolumeSnapshotClass`. Contribuiscono a definire più classi di storage e sono referenziati dagli snapshot dei gruppi di volumi per associare lo snapshot alla classe di snapshot richiesta. Ogni snapshot del gruppo di volumi è associato a una singola classe di snapshot del gruppo di volumi.

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

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

IL `driver` specifica a Kubernetes che le richieste per gli snapshot del gruppo di volumi del `csi-group-snap-class` la classe è gestita da Trident. La `deletionPolicy` specifica l'azione da intraprendere quando uno snapshot di gruppo deve essere eliminato. Quando `deletionPolicy` è impostato su `Delete` , gli oggetti snapshot del gruppo di volumi e lo snapshot sottostante sul cluster di archiviazione vengono rimossi quando uno snapshot viene eliminato. In alternativa, impostarlo su `Retain` significa che `VolumeGroupSnapshotContent` e lo snapshot fisico vengono conservati.

Oggetti Kubernetes `VolumeGroupSnapshot`

Un Kubernetes `VolumeGroupSnapshot` Un oggetto è una richiesta di creazione di uno snapshot di più volumi. Proprio come un PVC rappresenta una richiesta effettuata da un utente per un volume, uno snapshot di un gruppo di volumi è una richiesta effettuata da un utente per creare uno snapshot di un PVC esistente.

Quando arriva una richiesta di snapshot del gruppo di volumi, Trident gestisce automaticamente la creazione dello snapshot del gruppo per i volumi sul backend ed espone lo snapshot creando un'istanza univoca `VolumeGroupSnapshotContent` oggetto. È possibile creare snapshot da PVC esistenti e utilizzarle come DataSource durante la creazione di nuovi PVC.

NOTA	Il ciclo di vita di un VolumeGroupSnapshot è indipendente dal PVC di origine: uno snapshot persiste anche dopo l'eliminazione 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. Lo snapshot del gruppo di volumi viene rimosso quando tutti gli snapshot associati vengono eliminati.

Oggetti Kubernetes `VolumeGroupSnapshotContent`

Un Kubernetes `VolumeGroupSnapshotContent` L'oggetto rappresenta uno snapshot di gruppo preso da un volume già provisionato. È analogo a e indica una `PersistentVolume` snapshot sottoposta a provisioning sul cluster di storage. Analogamente agli `PersistentVolumeClaim` oggetti e `PersistentVolume`, quando viene creato uno snapshot, l' `VolumeSnapshotContent`oggetto mantiene una mappatura uno a uno all' `VolumeSnapshot`oggetto, che aveva richiesto la creazione dello snapshot.

IL `VolumeGroupSnapshotContent` l'oggetto contiene dettagli che identificano il gruppo di snapshot, come ad esempio `volumeGroupSnapshotHandle` e singoli volumiSnapshotHandles esistenti sul sistema di archiviazione.

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

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.

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

I CRD sono un costrutto Kubernetes. Gli oggetti delle risorse sopra definite vengono creati da Trident. Come semplice esempio, quando 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.

OggettiTrident `StorageClass`

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.

NOTA	Con Kubernetes, questi oggetti vengono creati automaticamente quando un Kubernetes `StorageClass` Che utilizza Trident come provisioner è registrato.

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

È possibile creare configurazioni delle classi di storage per definire direttamente le classi di storage utilizzando l'API REST. Tuttavia, per le implementazioni di Kubernetes, ci aspettiamo che vengano create al momento della registrazione dei nuovi Kubernetes `StorageClass` oggetti.

Oggetti backend Trident

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

NOTA	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".

OggettiTrident `StoragePool`

I pool di archiviazione rappresentano le diverse posizioni disponibili per il provisioning su ciascun backend. Per ONTAP, questi corrispondono agli aggregati nelle SVM. Per NetApp HCI/ SolidFire, questi corrispondono alle bande QoS specificate dall'amministratore. Ogni pool di archiviazione ha un set di attributi di archiviazione distinti, che ne definiscono le caratteristiche prestazionali e di protezione dei dati.

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

OggettiTrident `Volume`

I volumi sono l'unità di provisioning di base, comprendente endpoint backend, come NFS share, e LUN iSCSI e FC. 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.

NOTA	• 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"
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
saltaCoda di Recupero	stringa	no	Durante l'eliminazione del volume, ignorare la coda di ripristino nell'archiviazione ed eliminare immediatamente il volume.

Trident genera `internalName` durante la creazione del volume. Si tratta di due fasi. Prima di tutto, prepende

il prefisso di storage (predefinito) `trident` o il prefisso nella configurazione back-end) al nome del volume, con conseguente nome del modulo `<prefix>-<volume-name>`. Quindi, procede alla cancellazione del nome, sostituendo i caratteri non consentiti nel backend. Per i backend ONTAP, sostituisce i trattini con i caratteri di sottolineatura (quindi, il nome interno diventa `<prefix>_<volume-name>`). Per i backend degli elementi, sostituisce i caratteri di sottolineatura con trattini.

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

OggettiTrident `Snapshot`

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

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

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

OggettoTrident `ResourceQuota`

Il deamonset Trident consuma una `system-node-critical` classe di priorità, la classe di priorità più elevata disponibile in Kubernetes, per garantire che Trident possa identificare e ripulire i volumi in fase di shutdown anomalo del nodo e consentire ai pod di daemonset Trident di prevenire i carichi di lavoro con una priorità più

bassa nei cluster in cui esiste una pressione elevata delle risorse.

A tale scopo, Trident utilizza un `ResourceQuota` oggetto per garantire che sia soddisfatta una classe di priorità "system-node-critical" sul daemonset Trident. Prima della distribuzione e della creazione di daemonset, Trident cerca l' `ResourceQuota`oggetto e, se non lo rileva, 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 si preferisce controllare la propria allocazione di risorse, è possibile rimuovere l'oggetto Trident `ResourceQuota` utilizzando 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, Trident abilita determinate autorizzazioni durante l'installazione. Nelle sezioni seguenti vengono descritte in dettaglio le autorizzazioni impostate da Trident.

NOTA	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 avviare `rpc-statd` NFS. Trident interroga i processi host per determinare se `rpc-statd` è in esecuzione prima di montare i 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 contenitori privilegiati; pertanto, non può essere abilitato in Trident.
SELinux	In OpenShift, i contenitori privilegiati vengono eseguiti nel `spc_t` dominio ("contenitore con privilegi speciali") e i contenitori senza privilegi vengono eseguiti nel `container_t` dominio. Su `containerd`, con `container-selinux` installato, tutti i contenitori vengono eseguiti nel `spc_t` dominio, il che disabilita effettivamente SELinux. Pertanto, Trident non aggiunge `seLinuxOptions` ai contenitori.
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.>`

ATTENZIONE 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`

Campo	Descrizione	Predefinito
`forbiddenSysctls`	No `sysctls` sono consentiti.	Vuoto
`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 di terze parti e le licenze utilizzate nel software NetApp per Trident nel file degli avvisi per ciascuna versione 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.