Documentazione Trident 26.02

Trident

NetApp
August 04, 2026

This PDF was generated from https://docs.netapp.com/it-it/trident-2602/index.html on August 04, 2026.
Always check docs.netapp.com for the latest.

Sommario

Documentazione Trident 26.02 . . . 1
Note di rilascio . . . 2
Novità . . . 2
Novità della versione 26.02.1 . . . 2
Novità della versione 26.02 . . . 3
Novità della versione 25.10 . . . 5
Modifiche in 25.06.2 . . . 7
Modifiche in 25.06.1 . . . 7
Modifiche in 25.06 . . . 8
Modifiche in 25.02.1 . . . 10
Modifiche in 25.02 . . . 10
Modifiche in 24.10.1 . . . 12
Modifiche in 24.10 . . . 12
Modifiche in 24.06 . . . 14
Modifiche in 24.02 . . . 15
Modifiche in 23.10 . . . 16
Modifiche in 23.07.1 . . . 16
Modifiche in 23.07 . . . 16
Modifiche in 23.04 . . . 17
Modifiche in 23.01.1 . . . 18
Modifiche in 23.01 . . . 19
Modifiche in 22.10 . . . 19
Modifiche in 22.07 . . . 21
Modifiche in 22.04 . . . 22
Modifiche in 22.01.1 . . . 22
Modifiche in 22.01.0 . . . 23
Modifiche in 21.10.1 . . . 23
Modifiche in 21.10.0 . . . 24
Problemi noti . . . 24
Trova ulteriori informazioni . . . 25
Versioni precedenti della documentazione . . . 26
NetApp Trident supporto per sistemi di storage ONTAP ASA r2 . . . 26
Operazioni supportate . . . 26
Operazioni non supportate . . . 27
Problemi noti . . . 27
VolumeSnapshots non raggiungono lo stato ReadyToUse . . . 27
Il ripristino dei backup Restic di file di grandi dimensioni può non riuscire . . . 27
Inizia . . . 28
Informazioni su Trident . . . 28
Informazioni su Trident . . . 28
Architettura di Trident . . . 29
Concetti . . . 32
Avvio rapido per Trident . . . 36

E ora? 37
Requisiti 37
Informazioni critiche su Trident 37
Frontend supportati (orchestratori) 37
Backend supportati (storage) 38
Supporto Trident per KubeVirt e OpenShift Virtualization 38
Requisiti delle funzionalità 39
Sistemi operativi host testati 39
Configurazione host 40
Configurazione del sistema storage 40
Porte Trident 40
Immagini dei container e versioni Kubernetes corrispondenti 40
Installare Trident 41
Scopri l'installazione di Trident 41
Informazioni critiche su Trident 26.02 41
Prima di iniziare 41
Scegli il tuo metodo di installazione 41
Scegli la modalità di installazione 44
Seleziona il processo in base al tuo metodo e modalità 44
Spostarsi tra i metodi di installazione 45
Altre opzioni di configurazione note 45
Installa utilizzando l'operatore Trident 45
Distribuisci manualmente l'operatore Trident (modalità standard) 45
Distribuisci manualmente l'operatore Trident (modalità offline) 51
Distribuisci l'operatore Trident tramite Helm (modalità standard) 57
Distribuire l'operatore Trident utilizzando Helm (Offline mode) 65
Personalizzare l'installazione dell'operatore Trident 73
Installare utilizzando tridentctl 85
Installare utilizzando tridentctl 85
Personalizza l'installazione di tridentctl 89
Installa utilizzando l'operatore certificato OpenShift 90
Installare Trident utilizzando OpenShift OperatorHub 90
Passa all'operatore Trident certificato OpenShift 93
Usa Trident 97
Prepara il nodo worker 97
Selezionare gli strumenti giusti 97
Rilevamento del servizio nodo 97
volumi NFS 98
volumi iSCSI 98
Volumi NVMe/TCP 102
SCSI over FC volumi 103
Prepararsi al provisioning dei volumi SMB 106
Configura e gestisci i backend 107
Configura i backend 107
Azure NetApp Files 108

Google Cloud NetApp Volumes . . . 127
Configura un backend NetApp HCI o SolidFire . . . 153
Driver SAN ONTAP . . . 158
Driver NAS ONTAP . . . 188
Amazon FSx for NetApp ONTAP . . . 225
Crea backend con kubectl . . . 262
Gestisci i backend . . . 268
Crea e gestisci classi di archiviazione . . . 279
Creare una storage class . . . 279
Gestisci le classi di archiviazione . . . 282
Effettua il provisioning e gestisci i volumi . . . 284
Effettua il provisioning di un volume . . . 284
Espandi volumi . . . 288
Comprendere i limiti del sottosistema RWX NVMe . . . 299
Scalabilità del controller . . . 300
Espansione automatica del volume . . . 305
Gestisci le policy di Autogrow . . . 311
Importa volumi . . . 320
Personalizza i nomi e le etichette dei volumi . . . 332
Condividere un volume NFS tra namespace . . . 335
Clona volumi tra namespace . . . 339
Replicare i volumi utilizzando SnapMirror . . . 342
Usa la topologia CSI . . . 348
Lavora con gli snapshot . . . 356
Lavorare con le Snapshot dei gruppi di volumi . . . 364
Gestisci e monitora Trident . . . 369
Aggiorna Trident . . . 369
Aggiorna Trident . . . 369
Aggiorna con l'operatore . . . 370
Aggiorna con tridentctl . . . 375
Gestisci Trident usando tridentctl . . . 375
Comandi e flag globali . . . 375
Opzioni e flag dei comandi . . . 377
Supporto plugin . . . 383
Monitorare Trident . . . 383
Panoramica . . . 383
Fase 1: definire un target Prometheus . . . 383
Fase 2: crea un Prometheus ServiceMonitor . . . 384
Passaggio 3: interrogare le metriche di Trident con PromQL . . . 386
Scopri la telemetria Trident AutoSupport . . . 387
Disabilita metriche Trident . . . 388
Disinstalla Trident . . . 388
Determinare il metodo di installazione originale . . . 388
Disinstallare un'installazione dell'operatore Trident . . . 388
Disinstalla un' `tridentctl`installazione . . . 389

Trident per Docker . . . 390
Prerequisiti per la distribuzione . . . 390
Verifica i requisiti . . . 390
Strumenti NVMe . . . 392
Strumenti FC . . . 393
Distribuisci Trident . . . 395
Metodo del plugin gestito da Docker (versione 1.13/17.03 e successive) . . . 395
Metodo tradizionale (versione 1.12 o precedente) . . . 397
Avvia Trident all'avvio del sistema . . . 398
Aggiorna o disinstalla Trident . . . 399
Aggiornamento . . . 399
Disinstallare . . . 401
Gestire i volumi . . . 401
Crea un volume . . . 401
Rimuovi un volume . . . 402
Clonare un volume . . . 402
Accedere ai volumi creati esternamente . . . 404
Opzioni di volume specifiche del driver . . . 404
Raccogli i registri . . . 408
Raccogli i registri per la risoluzione dei problemi . . . 409
Suggerimenti generali per la risoluzione dei problemi . . . 409
Gestisci più istanze di Trident . . . 410
Passaggi per il plugin gestito da Docker (versione 1.13/17.03 o successiva) . . . 410
Passaggi per il tradizionale (versione 1.12 o precedente) . . . 411
Opzioni di configurazione dello storage . . . 411
Opzioni di configurazione globale . . . 411
Configurazione ONTAP . . . 412
Configurazione software Element . . . 420
Problemi e limitazioni noti . . . 422
L'aggiornamento di Trident Docker Volume Plugin alla versione 20.10 e successive da versioni precedenti provoca un errore di aggiornamento con l'errore no such file or directory. . . . 422
I nomi dei volumi devono essere lunghi almeno 2 caratteri. . . . 423
Docker Swarm ha alcuni comportamenti che impediscono a Trident di supportarlo con ogni combinazione di storage e driver. . . . 423
Se è in corso 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 in fase di provisioning. . . . 423
Buone pratiche e raccomandazioni . . . 424
Distribuzione . . . 424
Distribuisci in uno spazio dei nomi dedicato . . . 424
Utilizza le quote e i limiti di intervallo per controllare il consumo di storage . . . 424
Configurazione dello storage . . . 424
Panoramica della piattaforma . . . 424
ONTAP e Cloud Volumes ONTAP best practices . . . 424
Best practice per SolidFire . . . 429

Dove trovare ulteriori informazioni? 431
Integra Trident 431
Selezione e distribuzione del driver 431
Progettazione della storage class 434
Progettazione di pool virtuali 435
Operazioni volume 436
Servizio metriche 439
Protezione dei dati e disaster recovery 440
Replica e recovery di Trident 440
Replica e recovery SVM 441
Replicazione e recovery del volume 442
Protezione dei dati Snapshot 442
Automatizzare il failover delle applicazioni stateful con Trident 443
Dettagli sul force detach 443
Dettagli sul failover automatico 444
Sicurezza 448
Sicurezza 448
Linux Unified Key Setup (LUKS) 449
Crittografia in volo Kerberos 456
Proteggi le applicazioni con Trident Protect 464
Scopri Trident Protect 464
E ora? 464
Installa Trident Protect 464
Requisiti di Trident Protect 464
Installa e configura Trident Protect 468
Installa il plugin Trident Protect CLI 472
Personalizza l'installazione di Trident Protect 476
Gestisci Trident Protect 481
Gestisci l'autorizzazione e il controllo degli accessi di Trident Protect 481
Monitorare le risorse di Trident Protect 488
Generare un bundle di supporto Trident Protect 493
Aggiorna Trident Protect 495
Gestire e proteggere le applicazioni 497
Utilizzare gli oggetti Trident Protect AppVault per gestire i bucket 497
Definisci un'applicazione per la gestione con Trident Protect 511
Proteggi le applicazioni utilizzando Trident Protect 516
Ripristina le applicazioni 529
Replicare le applicazioni utilizzando NetApp SnapMirror e Trident Protect 547
Migra le applicazioni utilizzando Trident Protect 564
Gestire gli hook di esecuzione di Trident Protect 568
Disinstallare Trident Protect 580
Blog di Trident e Trident Protect 581
Blog di Trident 581
Blog di Trident Protect 581
Conoscenza e supporto 583

Domande frequenti . . . 583
Domande generali . . . 583
Installa e usa Trident su un cluster Kubernetes . . . 583
Risoluzione dei problemi e supporto . . . 584
Aggiorna Trident . . . 585
Gestisci backend e volumi . . . 586
Risoluzione dei problemi . . . 590
Risoluzione dei problemi generali . . . 590
Distribuzione di Trident non riuscita utilizzando l'operatore . . . 591
Distribuzione di Trident non riuscita utilizzando `tridentctl` . . . 593
Rimuovere completamente Trident e CRDs . . . 593
Errore di destaging del nodo NVMe con namespace di blocchi raw RWX su Kubernetes 1.26 . . . 594
I client NFSv4.2 segnalano "argomento non valido" dopo l'aggiornamento ONTAP quando si prevede che "v4.2-xattrs" sia abilitato . . . 595
Supporto . . . 595
Ciclo di vita del supporto Trident . . . 595
Auto-supporto . . . 596
Supporto della community . . . 596
NetApp supporto tecnico . . . 596
Per ulteriori informazioni . . . 596
Riferimento . . . 597
Porte Trident . . . 597
Panoramica . . . 597
API REST Trident . . . 599
Quando utilizzare l'API REST . . . 599
Utilizzo delle REST API . . . 599
Opzioni della riga di comando . . . 600
Registrazione . . . 600
Kubernetes . . . 600
Docker . . . 601
REST . . . 601
Oggetti Kubernetes e Trident . . . 601
Come interagiscono gli oggetti tra loro? . . . 601
Oggetti `PersistentVolumeClaim` Kubernetes . . . 602
Oggetti `PersistentVolume` Kubernetes . . . 604
Oggetti `StorageClass` Kubernetes . . . 604
Oggetti `VolumeSnapshotClass` Kubernetes . . . 608
Oggetti `VolumeSnapshot` Kubernetes . . . 608
Oggetti `VolumeSnapshotContent` Kubernetes . . . 608
Oggetti `VolumeGroupSnapshotClass` Kubernetes . . . 609
Oggetti `VolumeGroupSnapshot` Kubernetes . . . 609
Oggetti `VolumeGroupSnapshotContent` Kubernetes . . . 610
Oggetti `CustomResourceDefinition` Kubernetes . . . 610
Trident `StorageClass` oggetti . . . 610

Oggetti backend Trident . . . 611
Trident `StoragePool` oggetti . . . 611
Trident `Volume` oggetti . . . 611
Trident `Snapshot` oggetti . . . 613
Oggetto Trident `ResourceQuota` . . . 613
Pod Security Standards (PSS) e Security Context Constraints (SCC) . . . 614
Contesto di sicurezza Kubernetes richiesto e campi correlati . . . 615
Standard di sicurezza dei Pod (PSS) . . . 615
Politiche di sicurezza dei Pod (PSP) . . . 616
Vincoli del contesto di sicurezza (SCC) . . . 617
Note legali . . . 620
Copyright . . . 620
Marchi . . . 620
Brevetti . . . 620
Informativa sulla privacy . . . 620
Open source . . . 620

Documentazione Trident 26.02

Note di rilascio

Novità

Le Release Notes forniscono informazioni sulle nuove funzionalità, i miglioramenti e le correzioni di bug nell'ultima versione di NetApp Trident.

> Il `tridentctl` binario per Linux fornito nel file zip dell'installatore è la versione testata e supportata. Tenere presente che il `macos` binario fornito nella `/extras` parte del file zip non è testato o supportato.

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. Consulta la "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. Consulta la "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. Consulta la "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. Consulta la "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. Consulta la "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. Consulta la "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. Consulta la "Documentazione sulla configurazione del backend di Azure NetApp Files" per ulteriori informazioni.
- Aggiunto il supporto per Kubernetes 1.35. Consulta 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 "Crea una pianificazione della protezione dei dati".
- 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, consultare "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

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

Trident

Miglioramenti

- **Kubernetes**:

- Aggiunto il supporto per CSI Volume Group Snapshots con le API Kubernetes Volume Group Snapshot v1beta1 per i driver ONTAP-NAS NFS e ONTAP-SAN-Economy, oltre a ONTAP-SAN (iSCSI e FC). Vedere "Lavorare con le Snapshot dei gruppi di volumi".
- Aggiunto il supporto per il failover automatico del carico di lavoro con force volume detach 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 "Automatizzare il failover delle applicazioni stateful con Trident".
- Maggiore concorrenza dei nodi Trident per una scalabilità superiore delle operazioni sui nodi per i volumi FCP.
- Aggiunto il supporto ONTAP AFX per il driver ONTAP NAS. Vedere "Opzioni ed esempi di configurazione NAS ONTAP".
- Aggiunto il supporto per la configurazione delle richieste e dei limiti delle risorse di CPU e memoria per i contenitori Trident tramite TridentOrchestrator CR e i valori del Helm chart. ("Issue #1000", "Issue #927", "Issue #853", "Issue #592", "Issue #110").
- Aggiunto il supporto FC per la personalità ASAr2. Vedere "Opzioni ed esempi di configurazione SAN ONTAP".
- Aggiunta un'opzione per servire le metriche Prometheus con HTTPS invece di HTTP. Vedere "Monitorare Trident".
- È stata 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".
- Il deployment di Trident ora viene eseguito con la classe di priorità system-cluster-critical.

- Aggiunta un'opzione per il controller Trident per utilizzare la rete host tramite helm, operator e tridentctl ("Issue #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 container node-driver-registrar CSI standardizzando Linux DaemonSet a node-driver-registrar per farlo corrispondere a Windows DaemonSet e alla denominazione dell'immagine del container.
  - Risolto un problema per cui le export policy per i qtree legacy non venivano aggiornate correttamente.
- **Openshift**:
  - Risolto il problema per cui il pod del nodo Trident non si avviava sui nodi Windows in Openshift perché SCC aveva allowHostDirVolumePlugin impostato su false ("Issue #950").
- Corretto il problema QPS dell'API Kubernetes che non veniva impostato tramite Helm ("Issue #975").
- Risolta l'impossibilità di montare un Persistent Volume Claim (PVC) basato su uno snapshot di un PVC con filesystem 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 backend (ad esempio, netappdvp_subsystem).
- Corretti errori di montaggio per volumi iSCSI durante l'aggiornamento di Trident da versioni precedenti alla 23.10 alla 24.10 e successive, risolvendo il problema "invalid SANType".
- Risolto il problema per cui lo stato del backend Trident non passava a online/offline senza riavviare il Trident controller.
- Risolta la 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 clone del volume.
- Risolto il problema di mancato unstaging del volume quando il percorso del dispositivo veniva modificato dal kernel.
- Risolto il problema di mancata rimozione dello stage del volume a causa del 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 di Trident.

Trident Protect

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

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 schedule supportate".
- Aggiunta un'annotazione alla schedule CR per configurare il timeout di bind del PVC, che verrà utilizzato dalla backup CR: `protect.trident.netapp.io/pvc-bind-timeout-sec`. Vedere "Annotazioni di backup e schedule supportate".
- Migliorati `tridentctl-protect` gli elenchi di backup e snapshot con un nuovo campo per indicare i fallimenti degli hook di esecuzione.

Modifiche in 25.06.2

Trident

Correzioni

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

Modifiche in 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 unmapppati dai sottosistemi.
  - Risolto un problema in cui più tentativi di chiudere un dispositivo LUKS portavano a errori nel detach dei volumi.
  - Corretto il destage del volume iSCSI quando il percorso del dispositivo è cambiato dalla sua creazione.
  - Blocca la clonazione a blocchi dei 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 backend SolidFire ("Issue #1008").

Modifiche in 25.06

Trident

Miglioramenti

- **Kubernetes**:
  - Aggiunto supporto per CSI Volume Group Snapshots con `v1beta1` Volume Group Snapshot Kubernetes API per il driver ONTAP-SAN iSCSI. Vedere "Lavorare con le Snapshot dei gruppi di volumi".

    > VolumeGroupSnapshot is a beta feature in Kubernetes con API beta. Kubernetes 1.32 è la versione minima richiesta per VolumeGroupSnapshot.

  - Aggiunto supporto per ONTAP ASA r2 per NVMe/TCP oltre a iSCSI. Vedere "Opzioni ed esempi di configurazione SAN ONTAP".
  - Aggiunto il supporto SMB sicuro per i volumi ONTAP-NAS e ONTAP-NAS-Economy. Ora è possibile utilizzare utenti e gruppi di Active Directory 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 i volumi iSCSI.
  - Aggiunto `--allow-discards` quando si aprono i volumi LUKS per consentire i comandi discard/TRIM per space reclamation.
  - Prestazioni migliorate durante la formattazione di volumi crittografati LUKS.
  - Pulizia LUKS migliorata per dispositivi LUKS non riusciti ma parzialmente formattati.
  - Idempotenza del nodo Trident migliorata per l'attacco e lo scollegamento del volume NVMe.
  - Aggiunto `internalID` field alla configurazione del volume Trident per il driver ONTAP-SAN-Economy.
  - Aggiunto supporto per la replicazione del volume con SnapMirror per backend NVMe. Vedere

"Replicare i volumi utilizzando SnapMirror".

Miglioramenti sperimentali

> Non utilizzare in ambienti di produzione.

- [Anteprima tecnica] Abilitate le operazioni simultanee del controller Trident tramite il `--enable-concurrency` feature flag. Ciò consente alle operazioni del controller di essere eseguite in parallelo, migliorando le prestazioni per 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).

- [Tech Preview] Aggiunto il supporto QOS manuale con il driver ANF.

Correzioni

- **Kubernetes**:
  - Risolto un problema con CSI NodeExpandVolume dove i dispositivi multipath potevano avere dimensioni incongruenti quando i dischi SCSI sottostanti non erano disponibili.
  - Risolto il problema della mancata pulizia delle policy di esportazione duplicate 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 ("Issue #831").
  - Corrette le policy di esportazione mancanti per i PVC creati da Snapshot ("Issue #1016").
  - Risolto il problema in cui le dimensioni del volume ANF non sono automaticamente allineate a incrementi di 1 GiB.
  - Risolto problema durante l'utilizzo di NFSv3 con Bottlerocket.
- Risolto il problema con i volumi ONTAP-NAS-Economy che si espandevano fino a 300 TB nonostante i fallimenti del ridimensionamento.
- Risolto il problema per cui le operazioni di clone split venivano eseguite in modo sincrono quando si utilizzava l'ONTAP REST API.

Deprecazioni:

- **Kubernetes**: aggiornato il supporto minimo di Kubernetes a v1.27.

Trident Protect

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

Miglioramenti

- Tempi di ripristino migliorati, offrendo la possibilità di eseguire backup completi più frequenti.

- Granularità migliorata della definizione dell'applicazione e ripristino selettivo con filtro Group-Version-Kind (GVK).
- Risincronizzazione efficiente e replicazione inversa quando si utilizza AppMirrorRelationship (AMR) con NetApp SnapMirror, per evitare la replicazione completa del PVC.
- Aggiunta la possibilità di utilizzare EKS Pod Identity per creare AppVault bucket, 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 agli snapshot 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.
- Aggiunta una convalida migliorata e messaggi di errore 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.
- KubeVirt VMSnapshots sono esclusi dalle operazioni di ripristino e failover per evitare errori.
- Risolto il problema con Kopia in cui gli snapshot venivano rimossi prematuramente perché la pianificazione di conservazione predefinita di Kopia sovrascriveva quanto impostato dall'utente nella pianificazione.

Modifiche in 25.02.1

Trident

Correzioni

- **Kubernetes**:
  - Risolto un problema nel trident-operator in cui i nomi e le versioni delle immagini sidecar venivano compilati in modo errato quando si utilizzava un registro immagini non predefinito ("Issue #983").
  - Risolto il problema per cui le sessioni multipath non riuscivano a ripristinarsi durante un ONTAP failover giveback ("Issue #961").

Modifiche in 25.02

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

Trident

Miglioramenti

- **Kubernetes**:
  - Aggiunto supporto per ONTAP ASA r2 per iSCSI.

    - Aggiunto il supporto per il force detach per i volumi ONTAP-NAS durante gli scenari di Non-Graceful Node Shutdown. I nuovi volumi ONTAP-NAS ora utilizzeranno policy di esportazione per volume gestite da Trident. Fornito un percorso di upgrade per i volumi esistenti per passare al nuovo modello di policy di esportazione durante l'annullamento della pubblicazione senza influire sui carichi di lavoro attivi.
    - Aggiunta annotazione cloneFromSnapshot.
    - Aggiunto supporto per la clonazione di volumi cross namespace.
    - Migliorate le risoluzioni di scansione auto-riparante iSCSI per avviare nuove scansioni in base all'esatto host, canale, target e ID LUN.
    - Aggiunto il supporto per Kubernetes 1.32.
- **OpenShift**:
    - Aggiunto il supporto per la preparazione automatica dei nodi iSCSI per RHCOS su cluster ROSA.
    - Aggiunto supporto per OpenShift Virtualization per i driver ONTAP.
- Aggiunto il supporto Fibre Channel sul driver ONTAP-SAN.
- Aggiunto il supporto NVMe LUKS.
- Passato all'immagine scratch per tutte le immagini di base.
- Aggiunta la rilevazione e la registrazione dello stato della connessione iSCSI quando le sessioni iSCSI dovrebbero essere registrate ma non lo sono ("Issue #961").
- Aggiunto supporto per volumi SMB con il driver google-cloud-netapp-volumes.
- Aggiunto supporto per consentire ai volumi ONTAP di saltare la coda di ripristino durante l'eliminazione.
- Aggiunto supporto per sovrascrivere le immagini predefinite utilizzando SHA invece dei tag.
- Aggiunto il flag image-pull-secrets all'installer tridentctl.

Correzioni

- **Kubernetes**:
    - Corretti gli indirizzi IP dei nodi mancanti dalle policy di esportazione automatica ("Issue #965").
    - Corretto il passaggio prematuro delle policy di esportazione automatiche alla policy per volume per ONTAP-NAS-Economy.
    - Corrette le credenziali di configurazione backend per supportare tutte le partizioni AWS ARN disponibili ("Issue #913").
    - Aggiunta opzione per disabilitare la riconciliazione del configuratore automatico nell'operatore Trident ("Issue #924").
    - Aggiunto securityContext per il container csi-resizer ("Issue #976").

Trident Protect

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

Miglioramenti

- Aggiunto il supporto per backup e ripristino delle VM di virtualizzazione KubeVirt / OpenShift per entrambi i volumeMode: File e volumeMode: Block (raw device) storage. Questo supporto è compatibile con tutti i

driver Trident e migliora le funzionalità di protezione esistenti durante la replica dello storage utilizzando NetApp SnapMirror con Trident Protect.

- Aggiunta la possibilità di controllare il comportamento di freeze 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 execution hook.
- Aggiunta la possibilità di configurare i vincoli di contesto di sicurezza (SCC) durante l'installazione di Trident Protect.
- Aggiunto supporto per la configurazione di nodeSelector durante l'installazione di Trident Protect.
- Aggiunto supporto per proxy HTTP / HTTPS in uscita per oggetti AppVault.
- Esteso ResourceFilter per consentire l'esclusione delle risorse con ambito cluster.
- Aggiunto supporto per il token di sessione AWS nelle credenziali S3 AppVault.
- Aggiunto il supporto per la raccolta delle risorse dopo gli hook di esecuzione pre-snapshot.

Correzioni

- Migliorata la gestione dei volumi temporanei per saltare la coda di ripristino del volume ONTAP.
- Le annotazioni SCC sono ora ripristinate ai valori originali.
- Efficienza di ripristino migliorata con supporto per operazioni parallele.
- Supporto migliorato per i timeout dell'hook di esecuzione per applicazioni di grandi dimensioni.

Modifiche in 24.10.1

Miglioramenti

- **Kubernetes**: Aggiunto supporto per Kubernetes 1.32.
- Aggiunta la rilevazione e la registrazione dello stato della connessione iSCSI quando le sessioni iSCSI dovrebbero essere registrate ma non lo sono ("Issue #961").

Correzioni

- Corretti gli indirizzi IP dei nodi mancanti dalle policy di esportazione automatica ("Issue #965").
- Corretto il passaggio prematuro delle policy di esportazione automatiche alla policy per volume per ONTAP-NAS-Economy.
- Aggiornate le dipendenze Trident e Trident-ASUP per risolvere CVE-2024-45337 e CVE-2024-45310.
- Rimosse le disconnessioni per i portali non-CHAP intermittenti non funzionanti durante l'auto-riparazione iSCSI ("Issue #961").

Modifiche in 24.10

Miglioramenti

- Il driver Google Cloud NetApp Volumes è ora generalmente disponibile per i volumi NFS e supporta il provisioning basato sulla zona.

- GCP Workload Identity verrà utilizzato come Cloud Identity 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.
- Ridotta la dimensione minima del volume di Azure NetApp Files a 50 GiB. La nuova dimensione minima di Azure dovrebbe essere generalmente disponibile a novembre.
- Aggiunto `denyNewVolumePools` parametro di configurazione per limitare i driver ONTAP-NAS-Economy e ONTAP-SAN-Economy ai pool Flexvol preesistenti.
- Aggiunto il rilevamento per l'aggiunta, la rimozione o la ridenominazione degli aggregati dall'SVM su tutti i driver ONTAP.
- Aggiunti 18 MiB di overhead ai LUKS LUN per garantire che le dimensioni PVC segnalate siano utilizzabili.
- Migliorata la gestione degli errori di stage e unstage dei nodi ONTAP-SAN e ONTAP-SAN-Economy per consentire a unstage di rimuovere i dispositivi dopo uno stage non riuscito.
- Aggiunto un generatore di ruoli personalizzato che consente ai clienti di creare un ruolo minimalista per Trident in ONTAP.
- Aggiunta registrazione aggiuntiva per la risoluzione dei problemi `lsscsi` ("Issue #792").

Kubernetes

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

  "Scopri di più su Trident Protect".
- Aggiunto un nuovo flag `--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 installer per la gestione automatica delle dipendenze del protocollo storage sui nodi del cluster Kubernetes. Compatibilità testata e verificata con Amazon Linux 2023 e il protocollo storage iSCSI
- Aggiunto supporto per il distacco forzato per volumi ONTAP-NAS-Economy durante scenari di Non-Graceful Node Shutdown.
- I nuovi volumi NFS ONTAP-NAS-Economy utilizzeranno policy di esportazione per-qtree quando si utilizza `autoExportPolicy` l'opzione backend. I qtree verranno mappati alle policy di esportazione restrittive del nodo solo al momento della pubblicazione, per migliorare il controllo degli accessi e la sicurezza. I qtree esistenti verranno convertiti al nuovo modello di policy di esportazione quando Trident annullerà la pubblicazione del volume da tutti i nodi, così da non influire sui carichi di lavoro attivi.
- Aggiunto il supporto per Kubernetes 1.31.

Miglioramenti sperimentali

- Aggiunta anteprima tecnica per il supporto Fibre Channel sul driver ONTAP-SAN.

Correzioni

- **Kubernetes**:
  - Corretto il webhook di ammissione Rancher che impediva le installazioni di Trident Helm ("Issue #839").
  - Chiave di Affinity fissa nei valori dell'helm chart ("Issue #898").
  - Corretto tridentControllerPluginNodeSelector/tridentNodePluginNodeSelector non funzionerà con il valore "true" ("Issue #899").
  - Eliminati gli snapshot effimeri creati durante la clonazione ("Issue #901").
- Aggiunto supporto per Windows Server 2019.
- Corretto `go mod tidy`in Trident repo ("Issue #767").

Deprecazioni

- **Kubernetes:**
  - Aggiornata la versione minima supportata di Kubernetes a 1.25.
  - Rimosso il supporto per POD Security Policy.

Rebranding del prodotto

A partire dalla versione 24.10, Astra Trident è stato rinominato Trident (NetApp Trident). Questo rebranding non influisce su alcuna funzionalità, piattaforma supportata o interoperabilità per Trident.

Modifiche in 24.06

Miglioramenti

- **IMPORTANTE**: Il `limitVolumeSize` parametro ora limita le dimensioni qtree/LUN nei driver ONTAP economy. Utilizza il nuovo `limitVolumePoolSize` parametro per controllare le dimensioni Flexvol in tali driver. ("Issue #341").
- Aggiunta la possibilità per l'auto-riparazione iSCSI di avviare scansioni SCSI tramite l'ID LUN esatto se sono in uso igroups obsoleti ("Issue #883").
- Aggiunto il supporto per le operazioni di clonazione e ridimensionamento del volume da consentire anche quando il backend è in modalità sospesa.
- Aggiunta la possibilità per le impostazioni di log configurate dall'utente per il controller Trident di essere propagate ai pod nodo Trident.
- Aggiunto il supporto in Trident per utilizzare REST per impostazione predefinita anziché ONTAPI (ZAPI) per le versioni ONTAP 9.15.1 e successive.
- Aggiunto supporto per nomi di volumi personalizzati e metadati sui backend di storage ONTAP per i nuovi volumi persistenti.
- Migliorato il `azure-netapp-files` (ANF) driver per abilitare automaticamente la directory snapshot per impostazione predefinita quando le opzioni di montaggio NFS sono impostate per utilizzare NFS versione 4.x.
- Aggiunto il supporto Bottlerocket per volumi NFS.
- Aggiunto il supporto dell'anteprima tecnica per Google Cloud NetApp Volumes.

Kubernetes

- Aggiunto il supporto per Kubernetes 1.30.
- Aggiunta la possibilità per Trident DaemonSet di pulire le mount zombie e i file di tracciamento residui all'avvio ("Issue #883").
- Aggiunta annotazione PVC `trident.netapp.io/luksEncryption` per l'importazione dinamica dei volumi LUKS ("Issue #849").
- Aggiunta la consapevolezza della topologia al driver ANF.
- Aggiunto il supporto per nodi Windows Server 2022.

Correzioni

- Risolti i problemi di installazione di Trident dovuti a transazioni obsolete.
- Corretto tridentctl per ignorare i messaggi di avviso da Kubernetes ("Issue #892").
- Modificata la priorità del controller Trident `SecurityContextConstraint` a `0` ("Issue #887").
- I driver ONTAP ora accettano dimensioni di volume inferiori a 20 MiB ("Problema[#885").
- Corretto Trident per impedire la riduzione dei volumi FlexVol durante l'operazione di ridimensionamento per il driver ONTAP-SAN.
- Risolto il problema di importazione del volume ANF con NFS v4.1.

Modifiche in 24.02

Miglioramenti

- Aggiunto supporto per Cloud Identity.
  - AKS con ANF - Azure Workload Identity verrà utilizzato come identità Cloud.
  - EKS con FSxN - il ruolo AWS IAM verrà utilizzato come identità cloud.
- Aggiunto supporto per installare Trident come componente aggiuntivo sul cluster EKS dalla console EKS.
- Aggiunta la possibilità di configurare e disabilitare l'auto-riparazione iSCSI ("Issue #864").
- Aggiunta la personalità Amazon FSx ai driver ONTAP per abilitare l'integrazione con AWS IAM e SecretsManager, e per consentire a Trident di eliminare i volumi FSx con backup ("Issue #453").

Kubernetes

- Aggiunto il supporto per Kubernetes 1.29.

Correzioni

- Messaggi di avviso ACP corretti, quando ACP non è abilitato ("Issue #866").
- Aggiunto un ritardo di 10 secondi prima di eseguire un clone split durante l'eliminazione dello snapshot per i driver ONTAP, quando un clone è associato allo snapshot.

Deprecazioni

- Rimosso il framework di attestazioni in-toto dai manifesti delle immagini multipiattaforma.

Modifiche in 23.10

Correzioni

- Espansione del volume fissa se una nuova dimensione richiesta è inferiore alla dimensione totale del volume per i driver di storage ontap-nas e ontap-nas-flexgroup ("Issue #834").
- Dimensioni del volume fisse per visualizzare solo le dimensioni utilizzabili del volume durante l'importazione per i driver di storage ontap-nas e ontap-nas-flexgroup ("Issue #722").
- Correzione della conversione del nome FlexVol per ONTAP-NAS-Economy.
- Risolto il problema di inizializzazione di Trident su un nodo Windows quando il nodo viene riavviato.

Miglioramenti

Kubernetes

Aggiunto il supporto per Kubernetes 1.28.

Trident

- Aggiunto supporto per l'utilizzo di Azure Managed Identities (AMI) con il driver di archiviazione azure-netapp-files.
- Aggiunto supporto per NVMe over TCP per il driver ONTAP-SAN.
- Aggiunta la possibilità di mettere in pausa il provisioning di un volume quando il backend è impostato sullo stato sospeso dall'utente ("Issue #558").

Modifiche in 23.07.1

Kubernetes: Corretta l'eliminazione del daemonset per supportare gli aggiornamenti senza tempi di inattività ("Issue #740").

Modifiche in 23.07

Correzioni

Kubernetes

- Corretto l'aggiornamento di Trident per ignorare i vecchi pod bloccati nello stato di terminazione ("Issue #740").
- Aggiunta tolleranza alla definizione "transient-trident-version-pod" ("Issue #795").

Trident

- Richieste ONTAPI (ZAPI) corrette per garantire che i numeri di serie LUN vengano interrogati durante l'ottenimento degli attributi LUN per identificare e correggere i dispositivi iSCSI fantasma durante le operazioni di Node Staging.
- Corretta la gestione degli errori nel codice del driver di archiviazione ("Issue #816").
- Risolto il problema di ridimensionamento della quota quando si utilizzano i driver ONTAP con use-rest=true.
- Correzione della creazione del clone del LUN in ontap-san-economy.

- Ripristina il campo delle informazioni di pubblicazione da `rawDevicePath` a `devicePath`; aggiunta logica per popolare e recuperare (in alcuni casi) `devicePath` campo.

Miglioramenti

Kubernetes

- Aggiunto supporto per l'importazione di snapshot pre-provisioned.
- Distribuzione e permessi Linux del daemonset ridotti al minimo ("Issue #817").

Trident

- Non viene più segnalato il campo di stato per volumi e snapshot "online".
- Aggiorna lo stato del backend se il backend ONTAP è offline ("Problemi #801", "#543").
- Il numero di serie del LUN viene sempre recuperato e pubblicato durante il workflow ControllerVolumePublish.
- Aggiunta logica aggiuntiva per verificare il numero di serie e la dimensione del dispositivo iSCSI multipath.
- Verifica aggiuntiva per i volumi iSCSI per garantire che il dispositivo multipath corretto venga rimosso dalla messa in scena.

Miglioramento sperimentale

Aggiunto il supporto in tech preview per NVMe over TCP per il driver ONTAP-SAN.

Documentazione

Sono stati apportati numerosi miglioramenti organizzativi e di formattazione.

Deprecazioni

Kubernetes

- Rimosso il supporto per le snapshot v1beta1.
- Rimosso il supporto per i volumi pre-CSI e le classi di archiviazione.
- Aggiornata la versione minima supportata di Kubernetes a 1.22.

Modifiche in 23.04

> Il distacco forzato del volume per i volumi ONTAP-SAN-* è supportato solo con le versioni di Kubernetes in cui è abilitato il gate della funzione Non-Graceful Node Shutdown. Il distacco forzato deve essere abilitato al momento dell'installazione utilizzando il flag dell'installer `--enable-force-detach` Trident.

Correzioni

- Corretto Trident Operator per utilizzare IPv6 localhost per l'installazione quando specificato in spec.
- Corrette le autorizzazioni del ruolo del cluster Trident Operator per essere in sincronia con le autorizzazioni del bundle ("Issue #799").
- Problema risolto con il collegamento del volume raw a blocchi su più nodi in modalità RWX.

- Corretto il supporto per la clonazione di FlexGroup e l'importazione di volumi SMB.
- Risolto il problema per cui il controller Trident non poteva spegnersi immediatamente ("Issue #811").
- Aggiunta correzione per elencare tutti i nomi di igroup associati a una LUN specificata con i driver ontap-san-*.
- Aggiunta una correzione per consentire ai processi esterni di essere eseguiti fino al completamento.
- Corretto errore di compilazione per l'architettura s390 ("Issue #537").
- Corretto il livello di registrazione errato durante le operazioni di montaggio dei volumi ("Issue #781").
- Corretto un potenziale errore di asserzione di tipo ("Issue #802").

Miglioramenti

- Kubernetes:
  - Aggiunto il supporto per Kubernetes 1.27.
  - Aggiunto il supporto per l'importazione di volumi LUKS.
  - Aggiunto il supporto per ReadWriteOncePod modalità di accesso al PVC.
  - Aggiunto il supporto per il force detach dei volumi ONTAP-SAN-* durante gli scenari di Non-Graceful Node Shutdown.
  - Tutti i volumi ONTAP-SAN-* utilizzeranno ora igroup per nodo. Le LUN saranno mappate su igroup solo quando saranno attivamente pubblicate su tali nodi per migliorare la nostra posizione di sicurezza. I volumi esistenti saranno opportunisticamente passati al nuovo schema di igroup quando Trident determinerà che è sicuro farlo senza impattare i carichi di lavoro attivi ("Issue #758").
  - Sicurezza di Trident migliorata grazie alla pulizia degli igroup gestiti da Trident non utilizzati dai backend ONTAP-SAN-*.
- Aggiunto il supporto per i volumi SMB con Amazon FSx ai driver di storage ontap-nas-economy e ontap-nas-flexgroup.
- Aggiunto il supporto per le condivisioni SMB con i driver di storage ontap-nas, ontap-nas-economy e ontap-nas-flexgroup.
- Aggiunto supporto per i nodi arm64 ("Issue #732").
- Migliorata la procedura di spegnimento di Trident disattivando prima i server API ("Issue #811").
- Aggiunto il supporto per la compilazione multipiattaforma per host Windows e arm64 al Makefile; vedi BUILD.md.

Deprecazioni

Kubernetes: Gli igroup backend-scoped non verranno più creati durante la configurazione dei driver ontap-san e ontap-san-economy ("Issue #758").

Modifiche in 23.01.1

Correzioni

- Corretto Trident Operator per utilizzare IPv6 localhost per l'installazione quando specificato in spec.
- Corretti i permessi del ruolo del cluster dell'operatore Trident per essere sincronizzati con i permessi del bundle "Issue #799".
- Aggiunta una correzione per consentire ai processi esterni di essere eseguiti fino al completamento.

- Problema risolto con il collegamento del volume raw a blocchi su più nodi in modalità RWX.
- Corretto il supporto per la clonazione di FlexGroup e l'importazione di volumi SMB.

Modifiche in 23.01

Kubernetes 1.27 è ora supportato in Trident. Si prega di aggiornare Trident prima di aggiornare Kubernetes.

Correzioni

- Kubernetes: Aggiunte opzioni per escludere la creazione di Pod Security Policy per risolvere le installazioni di Trident tramite Helm ("Problemi #783, #794").

Miglioramenti

Kubernetes

- Aggiunto il supporto per Kubernetes 1.26.
- Miglioramento dell'utilizzo complessivo delle risorse Trident RBAC ("Issue #757").
- Aggiunta l'automazione per rilevare e correggere le sessioni iSCSI interrotte o stantie sui nodi host.
- Aggiunto il supporto per l'espansione dei volumi LUKS criptati.
- Kubernetes: Aggiunto il supporto per la rotazione delle credenziali per i volumi crittografati LUKS.

Trident

- Aggiunto il supporto per i volumi SMB con Amazon FSx for NetApp ONTAP al driver di archiviazione ontap-nas.
- Aggiunto il supporto per le autorizzazioni NTFS quando si utilizzano volumi SMB.
- Aggiunto supporto per i pool di storage per i volumi GCP con livello di servizio CVS.
- Aggiunto il supporto per l'uso opzionale di flexgroupAggregateList quando si creano FlexGroups con il driver di storage ontap-nas-flexgroup.
- Prestazioni migliorate per il driver di archiviazione ontap-nas-economy quando si gestiscono più FlexVol volumi
- Abilitati gli aggiornamenti dataLIF per tutti i driver di storage ONTAP NAS.
- Aggiornata la convenzione di naming per il deployment di Trident e DaemonSet per riflettere il sistema operativo del nodo host.

Deprecazioni

- Kubernetes: Aggiornato il minimo supportato di Kubernetes a 1.21.
- I DataLIF non devono più essere specificati quando si configurano `ontap-san` o `ontap-san-economy` driver.

Modifiche in 22.10

È necessario leggere le seguenti informazioni critiche prima di eseguire l'aggiornamento a Trident 22.10.

Informazioni critiche su Trident 22.10

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

  L'utilizzo di una configurazione non multipath o l'utilizzo di `find_multipaths: yes` o `find_multipaths: smart` nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'utilizzo di `find_multipaths: no` dalla release 21.07.

Correzioni

- Corretto un problema specifico del backend ONTAP creato utilizzando il campo `credentials` che non riesce ad andare online durante l'aggiornamento 22.07.0 ("Issue #759").
- **Docker:** Risolto un problema che causava il mancato avvio del plugin del volume Docker in alcuni ambienti ("Issue #548" e "Issue #760").
- Corretto il problema SLM specifico per ONTAP SAN backends per garantire che solo il sottoinsieme di dataLIF appartenenti ai nodi di reporting venga pubblicato.
- Risolto un problema di prestazioni in cui si verificavano scansioni non necessarie per iSCSI LUN durante l'associazione di un volume.
- Rimossi i tentativi granulari all'interno del flusso di lavoro Trident iSCSI per fallire rapidamente e ridurre gli intervalli di tentativi esterni.
- Risolto il problema per cui veniva restituito un errore durante il flushing di un dispositivo iSCSI quando il dispositivo multipath corrispondente era già stato flushato.

Miglioramenti

- Kubernetes:
  - Aggiunto il supporto per Kubernetes 1.25. Devi aggiornare Trident a 22.10 prima di aggiornare a Kubernetes 1.25.
  - Aggiunti un ServiceAccount, un ClusterRole e un ClusterRoleBinding separati per il Trident Deployment e DaemonSet per consentire futuri miglioramenti dei permessi.
  - Aggiunto supporto per "condivisione di volumi cross-namespace".
- Tutti i driver di storage Trident `ontap-*` ora funzionano con l'ONTAP REST API.
- Aggiunto nuovo operator yaml (`bundle_post_1_25.yaml`) senza un `PodSecurityPolicy` per supportare Kubernetes 1.25.
- Aggiunto "supporto per volumi crittografati LUKS" per `ontap-san` e `ontap-san-economy` driver di archiviazione.
- Aggiunto il supporto per nodi Windows Server 2019.
- Aggiunto "supporto per volumi SMB sui nodi Windows" attraverso il `azure-netapp-files` driver di archiviazione.
- Il rilevamento automatico della commutazione MetroCluster per i driver ONTAP è ora generalmente disponibile.

Deprecazioni

- **Kubernetes:** Aggiornato il supporto minimo di Kubernetes alla 1.20.
- Rimosso il driver Astra Data Store (ADS).
- Rimosso il supporto per `yes` e `smart` opzioni per `find_multipaths` quando si configura il multipathing dei nodi worker per iSCSI.

Modifiche in 22.07

Correzioni

Kubernetes

- Corretto il problema relativo alla gestione dei valori booleani e numerici per il selettore dei nodi quando si configura Trident con Helm o il Trident Operator. ("Issue GitHub #700")
- Corretto il problema nella gestione degli errori da percorsi non-CHAP, in modo che kubelet ritenti se fallisce. "Issue GitHub #736")

Miglioramenti

- Passaggio da k8s.gcr.io a registry.k8s.io come registro predefinito per le immagini CSI
- I volumi ONTAP-SAN ora utilizzeranno igroup per nodo e mapperanno le LUN agli igroup solo durante la pubblicazione attiva su tali nodi per migliorare la nostra sicurezza. I volumi esistenti verranno opportunisticamente convertiti al nuovo schema igroup quando Trident determinerà che è sicuro farlo senza influire sui carichi di lavoro attivi.
- Incluso un ResourceQuota con le installazioni di Trident per garantire che Trident DaemonSet sia programmato quando il consumo di PriorityClass è limitato per impostazione predefinita.
- Aggiunto il supporto per le funzionalità di rete al driver Azure NetApp Files. ("Issue GitHub #717")
- Aggiunta l'anteprima tecnica del rilevamento automatico del passaggio MetroCluster ai driver ONTAP. ("Issue GitHub #228")

Deprecazioni

- **Kubernetes:** Aggiornato il supporto minimo di Kubernetes alla 1.19.
- La configurazione del backend non consente più tipi di autenticazione multipli in una singola configurazione.

Rimozioni

- Il driver AWS CVS (deprecato dalla 22.04) è stato rimosso.
- Kubernetes
  - Rimossa la capability SYS_ADMIN non necessaria dai pod dei nodi.
  - Riduce nodeprep a semplici informazioni sull'host e alla scoperta di servizi attivi per effettuare una conferma best-effort che i servizi NFS/iSCSI siano disponibili sui nodi worker.

Documentazione

È stata aggiunta una nuova sezione "Standard di sicurezza del pod" (PSS) che descrive in dettaglio le autorizzazioni abilitate da Trident durante l'installazione.

Modifiche in 22.04

NetApp sta migliorando e perfezionando continuamente i suoi prodotti e servizi. Ecco alcune delle ultime funzionalità di Trident. Per le versioni precedenti, fare riferimento a "Versioni precedenti della documentazione".

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

Correzioni

- Analisi migliorata dei nomi degli iSCSI initiator. ("Issue GitHub #681")
- Risolto il problema per cui i parametri della classe di archiviazione CSI non erano consentiti. ("Issue GitHub #598")
- Corretta la dichiarazione di chiave duplicata in Trident CRD. ("Issue GitHub #671")
- Corretto il registro Snapshot CSI impreciso. ("Issue GitHub #629"))
- Risolto il problema relativo all'annullamento della pubblicazione dei volumi sui nodi eliminati. ("Issue GitHub #691")
- Aggiunta la gestione delle incongruenze del file system sui dispositivi a blocchi. ("Issue GitHub #656")
- Risolto il problema relativo all'estrazione delle immagini di auto-supporto quando si imposta il `imageRegistry` flag durante l'installazione. ("Issue GitHub #715")
- Risolto il problema per cui 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. ("Issue GitHub #698")
- Trident ora aggiunge intestazioni HSTS alle risposte dai suoi endpoint sicuri.
- Trident ora tenta di abilitare automaticamente la funzionalità di autorizzazioni Unix di Azure NetApp Files.
- **Kubernetes**: Il daemonset Trident ora viene eseguito con la classe di priorità system-node-critical. ("Issue GitHub #694")

Rimozioni

Il driver E-Series (disattivato dalla versione 20.07) è stato rimosso.

Modifiche in 22.01.1

Correzioni

- Risolto il problema relativo all'annullamento della pubblicazione dei volumi sui nodi eliminati. ("Issue GitHub #691")
- Risolto il panic durante l'accesso ai campi nil per lo spazio aggregato nelle risposte API ONTAP.

Modifiche in 22.01.0

Correzioni

- **Kubernetes:** Aumenta il tempo di ripetizione del backoff della 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.
- I DataLIF ONTAP SAN IPv6 ora funzionano se specificati tra parentesi.
- Risolto il problema per cui il tentativo di importare un volume già importato restituiva EOF lasciando PVC nello stato in sospeso. ("Issue GitHub #489")
- Risolto il problema per cui le prestazioni di Trident rallentano quando vengono creati più di 32 snapshot su un SolidFire volume.
- 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 Kubernetes:
  - Aggiunto il supporto per Kubernetes 1.23.
  - Aggiungere opzioni di pianificazione per i pod Trident quando installati tramite Trident Operator o Helm. ("Issue GitHub #651")
- Consenti volumi tra regioni nel driver GCP. ("Issue GitHub #633")
- Aggiunto supporto per l'opzione 'unixPermissions' ai volumi di Azure NetApp Files. ("Issue GitHub #666")

Deprecazioni

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

Modifiche in 21.10.1

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

Correzioni

- Risolta una potenziale condizione di competizione durante l'importazione di un volume su un backend GCP CVS che poteva causare il fallimento dell'importazione.
- Risolto un problema che può mettere il Trident controller in uno stato di CrashLoopBackOff quando un nodo viene rimosso e poi aggiunto nuovamente al cluster Kubernetes (GitHub issue 669).
- Risolto il problema per cui le SVM non venivano più rilevate se non veniva specificato alcun nome SVM (GitHub issue 612).

Modifiche in 21.10.0

Correzioni

- Risolto il problema per cui i cloni dei volumi XFS non potevano essere montati sullo stesso nodo del volume di origine (GitHub issue 514).
- Risolto il problema per cui Trident registrava un errore fatale durante l'arresto (GitHub issue 597).
- Correzioni relative a Kubernetes:
  - Restituisce lo spazio utilizzato da un volume come minimo restoreSize quando si creano snapshot con `ontap-nas` e `ontap-nas-flexgroup` driver (GitHub issue 645).
  - Risolto il problema per cui `Failed to expand filesystem` errore veniva registrato dopo il ridimensionamento del volume (GitHub issue 560).
  - Risolto il problema per cui un pod poteva rimanere bloccato in `Terminating` stato (GitHub issue 572).
  - Risolto il caso in cui un `ontap-san-economy` FlexVol poteva essere pieno di LUN snapshot (GitHub issue 533).
  - Risolto il problema dell'installer YAML personalizzato con un'immagine diversa (GitHub issue 613).
  - Corretto il calcolo delle dimensioni dello snapshot (GitHub issue 611).
  - Risolto il problema per cui tutti gli installer Trident potevano identificare Kubernetes semplice come OpenShift (GitHub issue 639).
  - Corretto l'operatore Trident per interrompere la riconciliazione se il server API Kubernetes non è raggiungibile (GitHub issue 599).

Miglioramenti

- Aggiunto supporto per `unixPermissions` option sui volumi GCP-CVS Performance.
- Aggiunto supporto per volumi CVS ottimizzati per la scalabilità in GCP nell'intervallo da 600 GiB a 1 TiB.
- Miglioramenti relativi a Kubernetes:
  - Aggiunto il supporto per Kubernetes 1.22.
  - Abilitato l'operatore Trident e il grafico Helm per funzionare con Kubernetes 1.22 (GitHub issue 628).
  - Aggiunta l'immagine dell'operatore al comando `tridentctl` images (GitHub issue 570).

Miglioramenti sperimentali

- Aggiunto il supporto per la replica del volume nel `ontap-san` driver.
- Aggiunto il supporto REST **anteprima tecnica** per i `ontap-nas-flexgroup`, `ontap-san` e `ontap-nas-economy` driver.

Problemi noti

I problemi noti identificano i problemi che potrebbero impedirti di utilizzare il prodotto con successo.

- Quando si aggiorna un cluster Kubernetes dalla versione 1.24 alla 1.25 o successiva che ha Trident installato, è necessario aggiornare values.yaml per impostare `excludePodSecurityPolicy` su `true` o aggiungere `--set excludePodSecurityPolicy=true` al comando `helm upgrade` prima di poter aggiornare il cluster.

- Trident ora impone un `fsType` (`fsType=""` vuoto per i volumi che non hanno il `fsType` specificato nel loro StorageClass. Quando si lavora con Kubernetes 1.17 o versioni successive, Trident supporta la fornitura di un `fsType` vuoto per i volumi NFS. Per i volumi iSCSI, è necessario impostare il `fsType` sul proprio StorageClass quando si applica un `fsGroup` utilizzando un Security Context.
- Quando si utilizza un backend su più istanze di Trident, ogni file di configurazione del backend dovrebbe avere un valore `storagePrefix` diverso per i backend ONTAP o utilizzare un `TenantName` diverso per i backend SolidFire. Trident non può rilevare i volumi che altre istanze di Trident hanno creato. Il tentativo di creare un volume esistente su backend ONTAP o SolidFire ha successo, perché Trident tratta la creazione del volume come un'operazione idempotente. Se `storagePrefix` o `TenantName` non sono diversi, potrebbero verificarsi collisioni di nomi per i volumi creati sullo stesso backend.
- Quando si installa Trident (utilizzando `tridentctl` o il Trident Operator) e si utilizza `tridentctl` per gestire Trident, è necessario assicurarsi che la variabile d'ambiente `KUBECONFIG` sia impostata. Questo è necessario per indicare il cluster Kubernetes su cui `tridentctl` dovrebbe lavorare. Quando si lavora con più ambienti Kubernetes, è necessario assicurarsi che il file `KUBECONFIG` sia correttamente referenziato.
- Per eseguire la space reclamation online per iSCSI PV, il sistema operativo sottostante sul nodo worker potrebbe richiedere che le opzioni di montaggio vengano passate al volume. Questo vale per le istanze RHEL/Red Hat Enterprise Linux CoreOS (RHCOS), che richiedono il `discard` "opzione di montaggio"; assicurati che l'opzione discard mountOption sia inclusa nel tuo [`StorageClass`^] per supportare l'online block discard.
- Se si dispone di più di un'istanza di Trident per cluster Kubernetes, Trident non può comunicare con altre istanze e non può rilevare altri volumi da esse creati, il che porta a un comportamento imprevisto e non corretto se più di un'istanza è in esecuzione all'interno di un cluster. Dovrebbe esserci una sola istanza di Trident per cluster Kubernetes.
- Se gli oggetti basati su `StorageClass`Trident vengono eliminati da Kubernetes mentre Trident è offline, Trident non rimuove le corrispondenti classi di archiviazione dal suo database quando torna online. Dovresti 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 supporto. È necessario rimuovere il volume tramite `tridentctl` o l'API REST.
- ONTAP non può fornire contemporaneamente più di un 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 del 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 rileva tutti i LIF iSCSI disponibili e li utilizza per stabilire la sessione multipath.

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

Trova ulteriori informazioni

- "Trident GitHub"
- "Blog di Trident"

Versioni precedenti della documentazione

Se non si utilizza Trident 26.02, la documentazione per le versioni precedenti è disponibile in base al "Ciclo di vita del supporto Trident".

- "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"

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. Consulta "Sistemi ASA r2" per maggiori 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 LUKS
- Replica di volumi SnapMirror
- Limitazione dell'utilizzo aggregato
- Modalità di prenotazione dello spazio
- Istantanee
- Tiering

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

Problemi noti

I problemi noti identificano le problematiche che potrebbero impedire il corretto utilizzo di questa versione del prodotto.

I seguenti problemi noti interessano la release corrente:

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 Restic di file di grandi dimensioni può non riuscire

Quando si ripristinano file da 30GB o più da un backup Amazon S3 effettuato con Restic, l'operazione di ripristino può non riuscire. Come soluzione alternativa, eseguire il backup dei dati utilizzando Kopia come data mover (Kopia è il data mover predefinito per i backup). Consulta "Proteggi le applicazioni utilizzando Trident Protect" per le istruzioni.

Inizia

Informazioni su Trident

Informazioni su Trident

Trident è un progetto open source completamente supportato e mantenuto da NetApp. È stato progettato per aiutarti a soddisfare le esigenze di persistenza delle tue applicazioni containerizzate utilizzando interfacce standard del settore, come la 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 orchestratore di storage dinamico conforme a Container Storage Interface (CSI) che si integra nativamente con "Kubernetes". Trident viene eseguito come un singolo Controller Pod più un Node Pod su ogni nodo worker nel cluster. Consultare "Architettura di Trident" per i dettagli.

Trident fornisce anche un'integrazione diretta con l'ecosistema Docker per le piattaforme di storage NetApp. Il NetApp Docker Volume Plugin (nDVP) supporta il provisioning e la gestione delle risorse di storage dalla piattaforma di storage agli host Docker. Consultare "Distribuisci Trident per Docker" per i dettagli.

> Se è la prima volta che si utilizza Kubernetes, è necessario familiarizzare con il "Concetti e strumenti 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 i prodotti NetApp

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

Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP" è un servizio AWS completamente gestito che consente di avviare 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 enterprise, alimentato da NetApp. Puoi eseguire i tuoi carichi di lavoro basati su file più impegnativi in Azure in modo nativo, con la performance e la ricca gestione 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 fornisce file storage dalle performance elevate, enterprise.

Software Element

"Elemento" consente all'amministratore dello storage di consolidare i carichi di lavoro garantendo le performance e consentendo un footprint di storage semplificato e snello.

NetApp HCI

"NetApp HCI" semplifica la gestione e la scalabilità del datacenter automatizzando le attività di routine e consentendo agli amministratori dell'infrastruttura di concentrarsi su funzioni più importanti.

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

NetApp ONTAP

"NetApp ONTAP" è il sistema operativo per lo storage multiprotocollo e unificato di NetApp che offre funzionalità avanzate di gestione dei dati per qualsiasi applicazione.

I sistemi ONTAP hanno configurazioni performance all-flash, ibride o all-HDD e offrono molti modelli di distribuzione: cluster on-premises FAS, AFA e ASA, ONTAP Select e Cloud Volumes ONTAP. Trident supporta questi modelli di distribuzione ONTAP.

Architettura di Trident

Trident viene eseguito come un singolo Controller Pod più un Node Pod su ogni nodo worker nel cluster. Il pod nodo deve essere in esecuzione su qualsiasi host in cui si desidera potenzialmente montare un volume Trident.

Comprendere i pod controller e i pod nodo

Trident si distribuisce come un singolo Pod del controller Trident e uno o più Pod dei nodi Trident sul cluster

Kubernetes e utilizza i *CSI Sidecar Containers* standard di Kubernetes per semplificare la distribuzione dei plugin CSI. "Container Sidecar CSI Kubernetes" sono mantenuti dalla comunità Kubernetes Storage.

Kubernetes "selettori di nodi" e "tolleranze e taint" sono utilizzati per vincolare un pod a essere eseguito su un nodo specifico o preferito. Puoi configurare i node selector e le toleration per i pod controller e node durante l'installazione di Trident.

- Il plugin del controller gestisce il provisioning e la gestione dei volumi, come snapshot e ridimensionamento.
- Il plugin del nodo gestisce il collegamento dello storage al nodo.

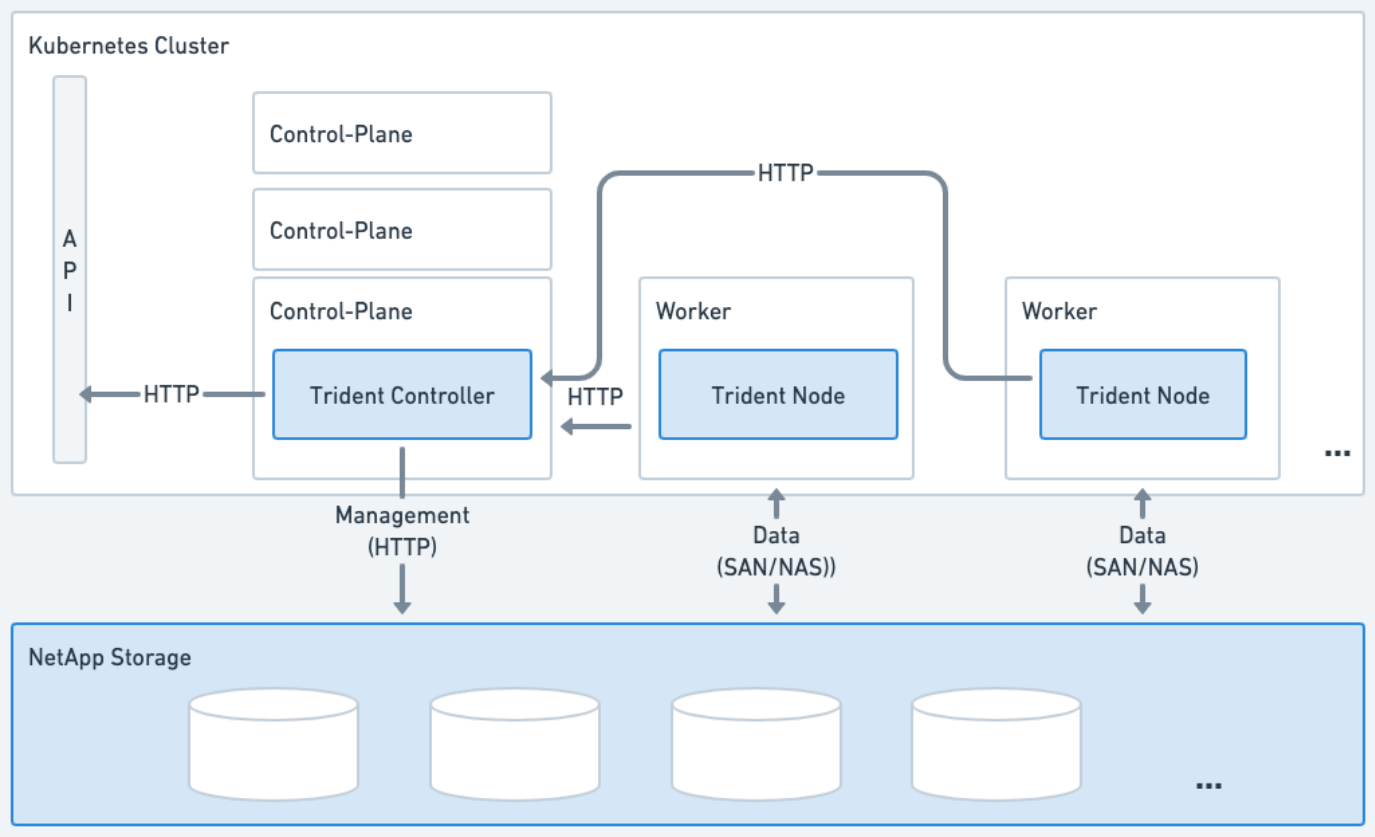


Figura 1. Trident distribuito sul cluster Kubernetes

Pod del controller Trident

Il Pod Controller Trident è un singolo Pod che esegue il plugin CSI Controller.

- Responsabile del provisioning e della gestione dei volumi nello storage NetApp
- Gestito da un Deployment Kubernetes
- Può essere eseguito sul control-plane o sui nodi worker, a seconda dei parametri di installazione.

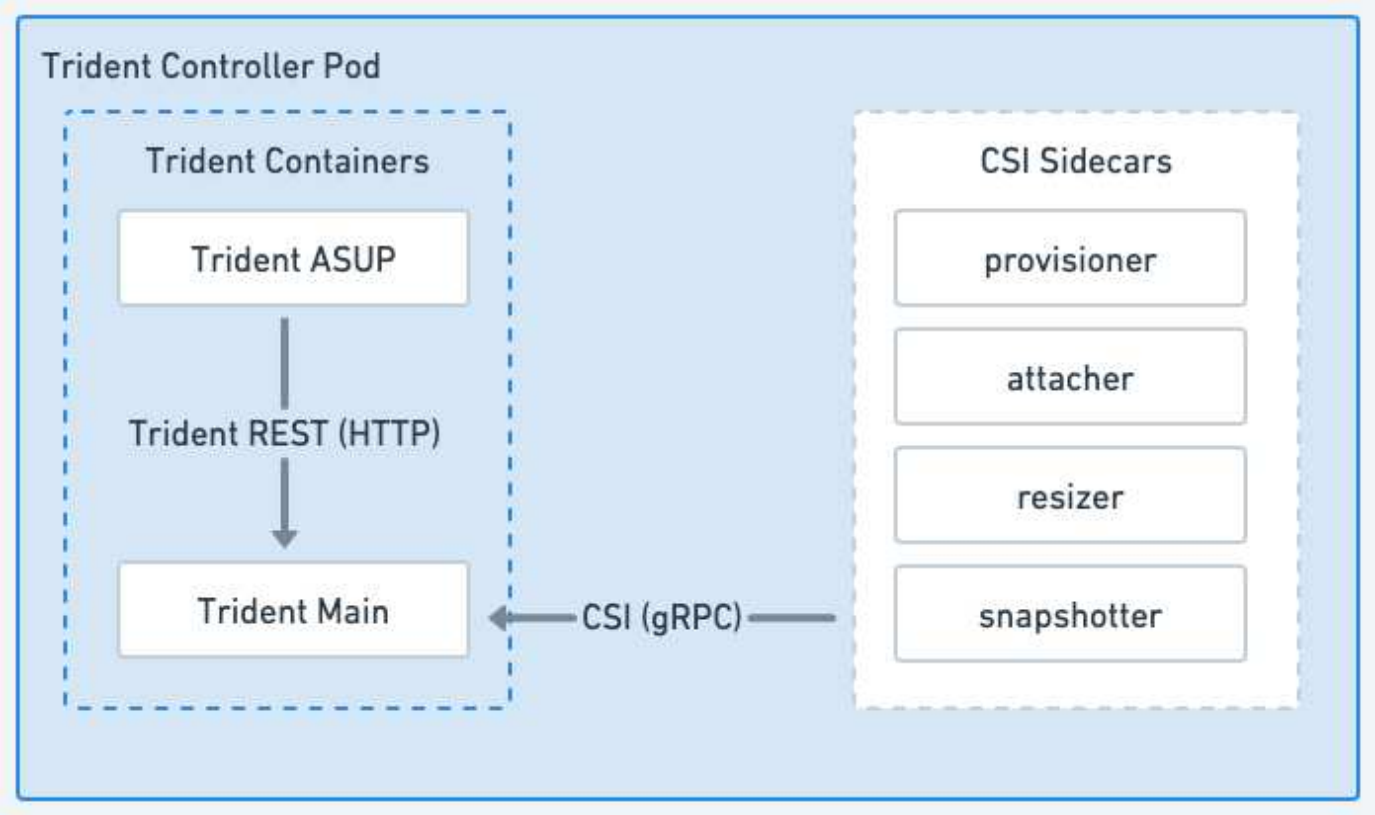


Figura 2. Diagramma del pod del controller Trident

Pod dei nodi Trident

I Pod Trident Node sono Pod privilegiati che eseguono il plugin CSI Node.

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

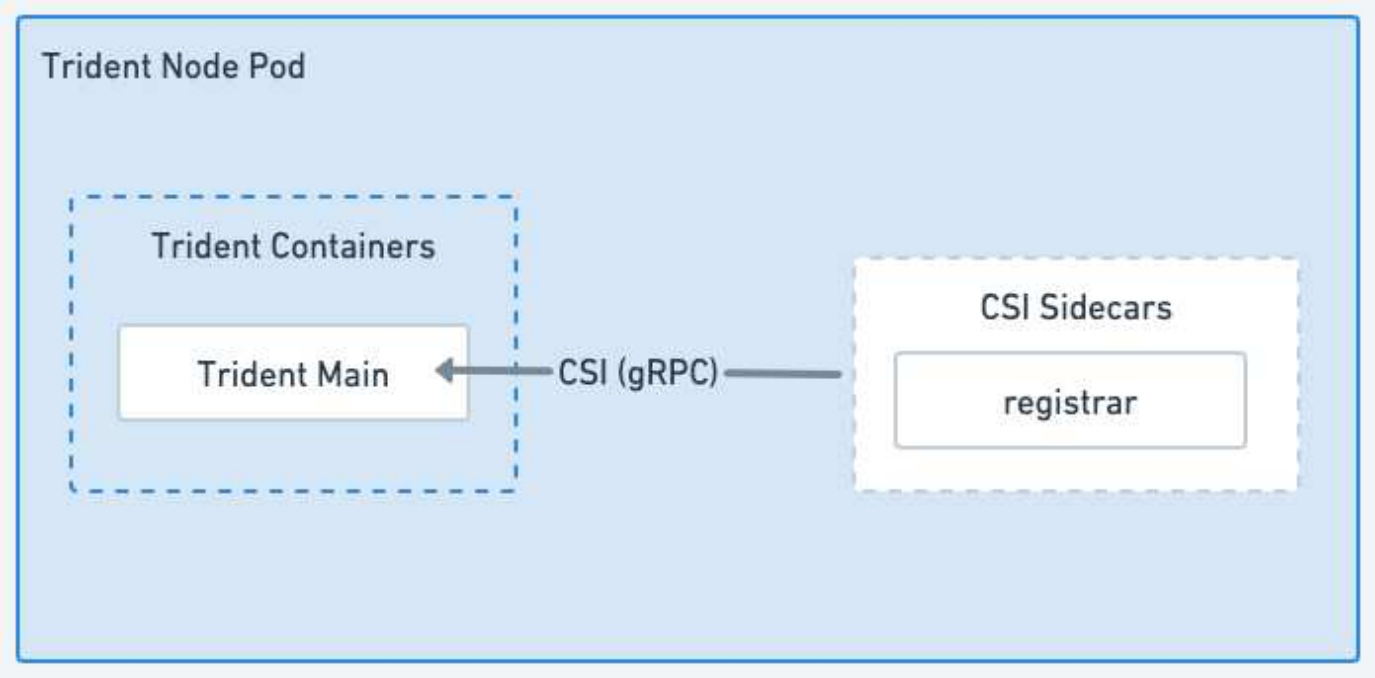


Figura 3. Diagramma del Node Pod di Trident

Architetture supportate del cluster Kubernetes

Trident è supportato con le seguenti architetture Kubernetes:

Architetture del cluster Kubernetes	Supportato	Installazione predefinita
Master singolo, calcolo	Sì	Sì
Master multipli, compute	Sì	Sì
Master, `etcd`, calcolo	Sì	Sì
Master, infrastruttura, compute	Sì	Sì

Concetti

Provisioning

Il provisioning in Trident si compone di due fasi principali. La prima fase associa una classe di storage al set di pool di storage back-end idonei e costituisce una necessaria preparazione prima del provisioning. La seconda fase include la creazione del volume vero e proprio e richiede la scelta di un pool di storage tra quelli associati alla classe di storage del volume in sospeso.

Associazione della storage class

L'associazione dei pool di storage back-end a una classe di storage si basa sia sugli attributi richiesti della classe di storage sia sui suoi `storagePools`, `additionalStoragePools` e `excludeStoragePools` elenchi. Quando crei una classe di storage, Trident confronta gli attributi e i pool offerti da ciascuno dei suoi

back-end 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 quel pool di storage all'insieme dei pool di storage idonei per quella classe di storage. Inoltre, Trident aggiunge tutti i pool di storage elencati nell' `additionalStoragePools`elenco a quell'insieme, anche se i loro attributi non soddisfano tutti o nessuno degli attributi richiesti dalla classe di storage. Dovresti usare l' `excludeStoragePools`elenco per escludere e rimuovere i pool di storage dall'utilizzo per una classe di storage. Trident esegue un processo simile ogni volta che aggiungi un nuovo back-end, verificando se i suoi pool di storage soddisfano quelli delle classi di storage esistenti e rimuovendo quelli contrassegnati come esclusi.

Creazione del volume

Trident utilizza quindi le associazioni tra classi di storage e pool di storage per determinare dove effettuare il provisioning dei volumi. Quando si crea un volume, Trident ottiene innanzitutto il set di pool di storage per la classe di storage di quel volume e, se si specifica un protocollo per il volume, Trident rimuove quei 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 ONTAP NAS non può fornire un volume basato su blocchi). Trident randomizza l'ordine di questo set risultante, per facilitare una distribuzione uniforme dei volumi, e quindi lo itera, tentando di effettuare il provisioning del volume su ciascun pool di storage a turno. Se riesce su uno, restituisce con successo, registrando eventuali errori riscontrati nel processo. Trident restituisce un errore **solo se** non riesce a effettuare il provisioning su **tutti** i pool di storage disponibili per la classe di storage e il protocollo richiesti.

Istantanee del volume

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

Scopri la creazione di snapshot del volume

- Per i `ontap-nas`, `ontap-san`, e `azure-netapp-files` driver, ogni Volume Persistente (PV) viene mappato a un FlexVol volume. Di conseguenza, gli snapshot del volume vengono creati come NetApp snapshot. La tecnologia snapshot NetApp offre maggiore stabilità, scalabilità, recuperabilità e prestazioni rispetto alle tecnologie snapshot concorrenti. Queste copie snapshot sono estremamente efficienti sia in termini di tempo necessario per crearle che di spazio di storage.
- Per il `ontap-nas-flexgroup` driver, ogni Persistent Volume (PV) è mappato a un FlexGroup. Di conseguenza, gli snapshot dei volumi vengono creati come snapshot NetApp FlexGroup. La tecnologia snapshot NetApp offre maggiore stabilità, scalabilità, recuperabilità e prestazioni rispetto alle tecnologie snapshot concorrenti. Queste copie snapshot sono estremamente efficienti sia in termini di tempo necessario per crearle che di spazio di storage.
- Per il `ontap-san-economy` driver, i PV sono mappati su LUN create su volumi FlexVol condivisi. Le VolumeSnapshots dei PV vengono ottenute eseguendo FlexClones della LUN associata. La tecnologia ONTAP FlexClone consente di creare copie anche dei dataset più grandi quasi istantaneamente. Le copie condividono blocchi di dati con i loro genitori, consumando solo lo spazio necessario per i metadati.
- Per il `solidfire-san` driver, ogni PV mappa a una LUN creata sul software NetApp Element/NetApp HCI cluster. VolumeSnapshots sono rappresentati da snapshot Element della LUN sottostante. Questi snapshot sono copie point-in-time e occupano solo una piccola quantità di risorse di sistema e spazio.
- Quando si lavora con i driver `ontap-nas` e `ontap-san`, le snapshot ONTAP sono copie point-in-time del FlexVol e consumano spazio sul FlexVol stesso. Questo può far sì che la quantità di spazio scrivibile nel volume si riduca con il tempo, man mano che vengono create o pianificate le snapshot. Un modo semplice per risolvere questo problema è aumentare il volume ridimensionandolo tramite Kubernetes. Un'altra opzione è eliminare le snapshot che non sono più necessarie. Quando una VolumeSnapshot creata tramite Kubernetes viene eliminata, Trident eliminerà la snapshot ONTAP associata. Le snapshot ONTAP che non

sono state create tramite Kubernetes possono anche essere eliminate.

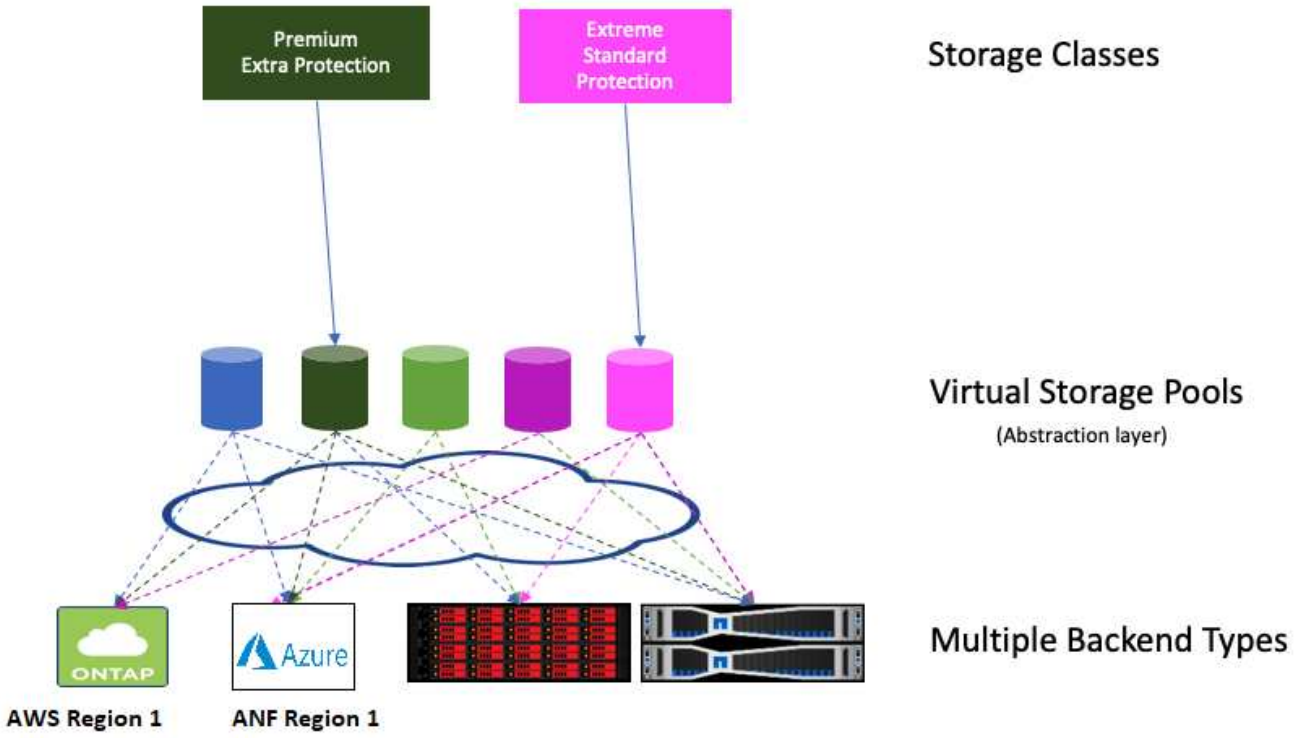
Con Trident, puoi utilizzare VolumeSnapshots per creare nuovi PV da esse. La creazione di PV da queste snapshot viene eseguita utilizzando la tecnologia FlexClone per i backend ONTAP supportati. Quando si crea un PV da una snapshot, il volume di supporto è un FlexClone del volume d'origine della snapshot. Il driver `solidfire-san` utilizza i cloni di volume del software Element per creare PV dalle snapshot. Qui crea un clone dalla snapshot Element.

Pool virtuali

I pool virtuali forniscono un livello di astrazione tra i backend di storage Trident e Kubernetes `StorageClasses`. Consentono a un amministratore di definire aspetti quali posizione, prestazioni e protezione per ciascun backend in modo comune e indipendente dal backend, senza che un `StorageClass` amministratore debba specificare quale backend fisico, pool di backend o tipo di backend utilizzare per soddisfare i criteri desiderati.

Informazioni sui pool virtuali

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



Qualsiasi aspetto specificato al di fuori dell'elenco dei virtual pool è globale per il backend e si applica a tutti i virtual pool, mentre ogni virtual pool potrebbe specificare uno o più aspetti individualmente (sovrascrivendo qualsiasi aspetto globale del backend).

- Quando si definiscono i pool virtuali, non tentare di riorganizzare l'ordine dei pool virtuali esistenti in una definizione di backend.
- Si sconsiglia di modificare gli attributi di un pool virtuale esistente. È necessario definire un nuovo pool virtuale per apportare modifiche.

La maggior parte degli aspetti è specificata in termini specifici per il backend. È fondamentale che i valori degli aspetti non siano esposti al di fuori del driver del backend e non siano disponibili per la corrispondenza in `StorageClasses`. Invece, l'amministratore definisce una o più etichette per ogni pool virtuale. Ogni etichetta è una coppia chiave:valore e le etichette possono essere comuni a diversi backend. Come gli aspetti, le etichette possono essere specificate per pool o globali per il backend. A differenza degli aspetti, che hanno nomi e valori predefiniti, l'amministratore ha piena discrezione nel definire le chiavi e i valori delle etichette secondo necessità. Per comodità, gli amministratori dello storage possono definire etichette per pool virtuale e raggruppare i volumi in base all'etichetta.

Le etichette dei pool virtuali possono essere definite utilizzando questi caratteri:

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

Un `StorageClass` identifica quale pool virtuale utilizzare facendo riferimento alle etichette all'interno di un parametro selector. I selettori dei pool virtuali supportano i seguenti operatori:

Operatore	Esempio	Il valore dell'etichetta di un pool deve:
`=`	performance=premium	Corrispondenza
`!=`	prestazioni!=extreme	Non corrisponde
`in`	posizione in (east, west)	Essere nell'insieme dei valori
`notin`	prestazioni notin (argento, bronzo)	Non essere nell'insieme dei valori
`<key>`	protezione	Esiste con qualsiasi valore
`!<key>`	!protezione	Non esiste

Gruppi di accesso al volume

Per saperne di più su come Trident utilizza "gruppi di accesso al volume".

Ignorare questa sezione se si utilizza CHAP, consigliata per semplificare la gestione ed evitare il limite di scala descritto di seguito. Inoltre, se si utilizza Trident in modalità CSI, è possibile ignorare questa sezione. Trident utilizza CHAP quando è installato come enhanced CSI provisioner.

Informazioni sui gruppi di accesso al volume

Trident può usare i gruppi di accesso ai volumi per controllare l'accesso ai volumi che fornisce. Se CHAP è disabilitata, si aspetta di trovare un gruppo di accesso chiamato `trident` a meno che non si specifichino uno

o più ID di gruppo di accesso nella configurazione.

Sebbene Trident associ i nuovi volumi ai gruppi di accesso configurati, non crea né gestisce i gruppi di accesso stessi. I gruppi di accesso devono esistere prima che il backend di storage venga aggiunto a Trident e devono contenere gli IQN iSCSI di ogni nodo nel cluster Kubernetes che potrebbe potenzialmente montare i volumi forniti da quel backend. Nella maggior parte delle installazioni, ciò include ogni nodo worker nel cluster.

Per i cluster Kubernetes con più di 64 nodi, è necessario utilizzare più gruppi di accesso. Ogni gruppo di accesso può contenere fino a 64 IQN e ogni volume può appartenere a quattro gruppi di accesso. Con il massimo di quattro gruppi di accesso configurati, qualsiasi nodo in un cluster fino a 256 nodi sarà in grado di accedere a qualsiasi volume. Per i limiti più recenti sui gruppi di accesso ai volumi, consultare "qui".

Se si sta modificando la configurazione da una che utilizza il gruppo di accesso predefinito `trident` a una che utilizza anche altri, includere l'ID del gruppo di accesso `trident` nell'elenco.

Avvio rapido per Trident

È possibile installare Trident e iniziare a gestire risorse di storage in pochi passi. Prima di iniziare, esaminare "Requisiti di Trident".

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

1 Preparare il nodo worker

Tutti i nodi worker nel cluster Kubernetes devono essere in grado di montare i volumi che hai fornito per i tuoi pod.

"Prepara il nodo worker"

2 Installare Trident

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

"Installare Trident"

3 Crea un backend

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

"Configura un backend" per il vostro sistema storage

4 Crea una StorageClass Kubernetes

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

"Creare una storage class"

5 Effettua il provisioning di un volume

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

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

"Effettua il provisioning di un volume"

E ora?

Ora puoi aggiungere altri backend, gestire classi di storage, gestire backend ed eseguire operazioni sui volumi.

Requisiti

Prima di installare Trident dovresti 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 critiche su Trident

- Kubernetes 1.36 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  L'utilizzo di una configurazione non multipath o l'utilizzo di `find_multipaths: yes` o `find_multipaths: smart` nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'utilizzo di `find_multipaths: no` dalla release 21.07.

Frontend supportati (orchestratori)

Trident supporta più motori di container e orchestratori, tra cui:

- Anthos On-Prem (VMware) e Anthos on bare metal 1.16
- Kubernetes 1.27 - 1.36
- OpenShift 4.12, 4.14 - 4.21 (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 la "Red Hat Extended Update Support (EUS) ciclo di vita", anche se si basano su versioni di Kubernetes che non sono più ufficialmente supportate upstream. Durante l'installazione di Trident in questi casi, puoi ignorare in tutta sicurezza eventuali messaggi di avviso relativi alla versione di Kubernetes.

- Rancher Kubernetes Engine 2 (RKE2) v1.28.x - 1.36.x

Trident funziona anche con una serie di altre offerte Kubernetes completamente gestite e autogestite, tra cui Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service (AKS), Mirantis Kubernetes Engine (MKE) e VMWare Tanzu Portfolio.

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

Prima di aggiornare un cluster Kubernetes dalla versione 1.25 alla 1.26 o successiva su cui è installato Trident, fare riferimento a "Aggiornare un'installazione di Helm".

Backend supportati (storage)

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

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

Supporto Trident per KubeVirt e OpenShift Virtualization

Driver di storage supportati:

Trident supporta i seguenti driver ONTAP per KubeVirt e OpenShift Virtualization:

- ontap-nas
- ontap-san (iSCSI, FCP, NVMe over TCP)
- ontap-san-economy (solo iSCSI)

Punti da considerare:

- Aggiornare la storage class in modo che abbia il `fsType` parametro (ad esempio: `fsType: "ext4"`) nell'ambiente OpenShift Virtualization. Se necessario, impostare la modalità volume su block in modo esplicito utilizzando il `volumeMode=Block` parametro nel `dataVolumeTemplates` per notificare a CDI di creare volumi di dati Block.
- *Modalità di accesso RWX per driver di storage a blocchi*: i driver ontap-san (iSCSI, NVMe/TCP, FC) e ontap-san-economy (iSCSI) sono supportati solo con "volumeMode: Block" (raw device). Per questi driver, il parametro `fstype` non può essere utilizzato perché i volumi sono forniti in modalità raw device.

- Per i flussi di lavoro di live-migration 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` (raw dispositivo)

Requisiti delle funzionalità

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

Caratteristica	Versione Kubernetes	Sono richiesti i feature gate?
Trident	1.27 - 1.36	No
Istantanee del volume	1.27 - 1.36	No
PVC da snapshot di volume	1.27 - 1.36	No
Ridimensionamento PV iSCSI	1.27 - 1.36	No
CHAP bidirezionale ONTAP	1.27 - 1.36	No
Politiche di esportazione dinamiche	1.27 - 1.36	No
Trident Operator	1.27 - 1.36	No
Topologia CSI	1.27 - 1.36	No

Sistemi operativi host testati

Sebbene Trident non supporti ufficialmente sistemi operativi specifici, i seguenti sono noti per funzionare:

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

  NVMe/TCP richiede RHEL 9 o versioni successive.

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

Per impostazione predefinita, Trident viene eseguito in un container e, pertanto, viene eseguito su qualsiasi worker Linux. Tuttavia, tali worker devono essere in grado di montare i volumi che Trident fornisce utilizzando il

client NFS standard o l'iSCSI initiator, a seconda dei backend utilizzati.

L' `tridentctl`utility funziona anche su una qualsiasi di queste distribuzioni di Linux.

Configurazione host

Tutti i nodi worker nel cluster Kubernetes devono essere in grado di montare i volumi che hai fornito per i tuoi pod. Per preparare i nodi worker, è necessario installare NFS, iSCSI o gli strumenti NVMe in base alla selezione del driver.

"Prepara il nodo worker"

Configurazione del sistema storage

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

"Configura i backend"

Porte Trident

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

"Porte Trident"

Immagini dei container e versioni Kubernetes corrispondenti

Per le installazioni air-gapped, il seguente elenco è un riferimento delle immagini container necessarie per installare Trident. Usa il comando `tridentctl images` per verificare l'elenco delle immagini container necessarie.

Immagini container necessarie per Trident 26.02

Versioni 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, v1.36.0	• docker.io/netapp/trident:26.02.0 • docker.io/netapp/trident-autosupport:26.02 • registry.k8s.io/sig-storage/csi-provisioner:v6.1.0 • registry.k8s.io/sig-storage/csi-attacher:v4.10.0 • registry.k8s.io/sig-storage/csi-resizer:v2.0.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.5.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.15.0 • docker.io/netapp/trident-operator:26.02.0 (opzionale)

Installare Trident

Scopri l'installazione di Trident

Per garantire che Trident possa essere installato in un'ampia varietà di ambienti e organizzazioni, NetApp offre diverse opzioni di installazione. Puoi installare Trident utilizzando l'operatore Trident (manualmente o tramite Helm) o con `tridentctl`. Questo argomento fornisce informazioni importanti per la scelta del processo di installazione più adatto a te.

Informazioni critiche su Trident 26.02

È necessario leggere le seguenti informazioni critiche su Trident.

Informazioni critiche su Trident

- Kubernetes 1.35 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  L'utilizzo di una configurazione non multipath o l'utilizzo di `find_multipaths: yes` o `find_multipaths: smart` nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'utilizzo 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.
- Capacità di montare volumi da tutti i nodi worker di Kubernetes.
- Un host Linux con `kubectl` (o `oc`, se si utilizza OpenShift) installato e configurato per gestire il cluster che si desidera utilizzare.
- La `KUBECONFIG`variabile di ambiente impostata per puntare alla configurazione del cluster Kubernetes.
- Se si utilizza Kubernetes con Docker Enterprise, "segui i loro passaggi per abilitare l'accesso CLI".
- Il cluster deve supportare carichi di lavoro privilegiati.

Se non hai familiarità con il "concetti di base", questo è il momento giusto per farlo.

Scegli il tuo metodo di installazione

Seleziona il metodo di installazione più adatto alle tue esigenze. Dovresti anche esaminare le considerazioni per "spostarsi tra i metodi" prima di prendere una decisione.

Utilizzo dell'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 persino "personalizza la distribuzione dell'operatore Trident" usando gli attributi nella `TridentOrchestrator` custom resource (CR).

I vantaggi dell'utilizzo dell'operatore Trident includono:

Creazione oggetto Trident

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

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

Responsabilità delle risorse

L'operatore Trident con ambito cluster gestisce le risorse associate a un'installazione Trident a livello di cluster. Questo riduce gli errori che potrebbero verificarsi durante la gestione delle risorse con ambito cluster utilizzando un operatore con ambito namespace. Questo è essenziale per l'auto-riparazione e l'applicazione di patch.

Capacità di self-healing

L'operatore monitora l'installazione di Trident e adotta misure attive per risolvere eventuali problemi, ad esempio quando il deployment viene eliminato o modificato accidentalmente. Un `trident-operator-<generated-id>` pod viene creato che associa una `TridentOrchestrator` CR a un'installazione di Trident. Questo garantisce che ci sia una sola istanza di Trident nel cluster e ne controlla la configurazione, assicurando che l'installazione sia idempotente. Quando vengono apportate modifiche all'installazione (ad esempio, l'eliminazione del deployment o del node daemonset), l'operatore le identifica e le corregge individualmente.

Aggiornamenti facili alle installazioni esistenti

È possibile aggiornare facilmente una distribuzione esistente con l'operatore. È sufficiente modificare il `TridentOrchestrator` CR per apportare aggiornamenti a un'installazione.

Ad esempio, considera uno scenario in cui è necessario abilitare Trident per generare log di debug. Per farlo, applica una patch al tuo `TridentOrchestrator` per impostare `spec.debug` su `true`:

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

Dopo `TridentOrchestrator` l'aggiornamento, l'operatore elabora gli aggiornamenti e applica le patch all'installazione esistente. Questo potrebbe innescare la creazione di nuovi pod per modificare l'installazione di conseguenza.

Reinstallazione pulita

L'operatore Trident con ambito cluster consente la rimozione pulita delle risorse con ambito cluster. Gli utenti possono disinstallare completamente Trident e reinstallarlo facilmente.

Gestione automatica degli upgrade di Kubernetes

Quando la versione Kubernetes del cluster viene aggiornata a 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.

Utilizzando `tridentctl`

Se hai una distribuzione esistente che deve essere aggiornata o se desideri personalizzarla in modo significativo, dovresti prendere in considerazione . Questo è il metodo convenzionale per distribuire Trident.

È possibile generare i manifest per le risorse Trident. Questo include la deployment, il daemonset, l'account di servizio e il cluster role che Trident crea come parte della sua installazione.

> A partire dalla versione 22.04, le chiavi AES non verranno più rigenerate ogni volta che si installa Trident. Con questa versione, Trident installerà un nuovo oggetto segreto che persiste tra le installazioni. Ciò significa che, `tridentctl` nella versione 22.04 è possibile disinstallare le versioni precedenti di Trident, ma le versioni precedenti non possono disinstallare le installazioni della 22.04. Selezionare il metodo di installazione appropriato.

Scegli la modalità di installazione

Determina il processo di distribuzione in base alla *modalità di installazione* (Standard, Offline o Remote) richiesta dalla tua 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 i registri predefiniti per memorizzare le immagini Trident (`docker.io`) e CSI (`registry.k8s.io`) richieste.

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

- Recupera le immagini del container tramite Internet
- Crea un deployment o un daemonset di nodi, che avvia i pod Trident su tutti i nodi idonei nel cluster Kubernetes

Installazione offline

La modalità di installazione offline potrebbe essere necessaria in una posizione isolata o sicura. In questo scenario, è possibile creare un singolo registro privato con mirroring o due registri con mirroring per archiviare le immagini Trident e CSI richieste.

Indipendentemente dalla configurazione del registro, le immagini CSI devono risiedere in un unico registro.

Installazione remota

Ecco una panoramica high-level del processo di installazione remota:

- Distribuisci la versione appropriata di `kubectl` sulla macchina remota da cui desideri distribuire Trident.
- Copia i file di configurazione dal cluster Kubernetes e imposta la `KUBECONFIG` variabile di ambiente sulla macchina remota.
- Avvia un `kubectl get nodes` comando per verificare che sia possibile connettersi al cluster Kubernetes richiesto.
- Completare la distribuzione dalla macchina remota utilizzando i passaggi di installazione standard.

Seleziona il processo in base al tuo metodo e modalità

Dopo aver preso le tue decisioni, seleziona il processo appropriato.

Metodo	Modalità di installazione
Trident operator (manualmente)	"Installazione standard" "Installazione offline"
Operatore Trident (Helm)	"Installazione standard" "Installazione offline"

Metodo	Modalità di installazione
`tridentctl`	"Installazione standard o offline"

Spostarsi tra i metodi di installazione

Puoi decidere di cambiare il metodo di installazione. Prima di farlo, considera quanto segue:

- Utilizzare sempre lo stesso metodo per installare e disinstallare Trident. Se si è eseguito il deployment con `tridentctl`, è necessario utilizzare la versione appropriata del `tridentctl` binario per disinstallare Trident. Analogamente, se si esegue il deployment con l'operatore, è necessario modificare la `TridentOrchestrator` CR e impostare `spec.uninstall=true` per disinstallare Trident.
- Se hai una distribuzione basata su operatore che vuoi rimuovere e usare invece `tridentctl` per distribuire Trident, dovresti prima modificare `TridentOrchestrator` e impostare `spec.uninstall=true` per disinstallare Trident. Quindi elimina `TridentOrchestrator` e la distribuzione dell'operatore. Puoi quindi 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 manualmente prima 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. Se non si esegue questa operazione, la distribuzione dell'operatore Trident basata su Helm non riuscirà a causa di un errore di convalida delle etichette e delle annotazioni.
- Se si dispone di una distribuzione basata su `tridentctl`, è possibile eseguire una distribuzione basata su Helm o su Operator senza disinstallare Trident.

Altre opzioni di configurazione note

Quando si installa Trident sui prodotti VMWare Tanzu Portfolio:

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

  La specifica della posizione del kubelet utilizzando `--kubelet-dir` è nota per funzionare con Trident Operator, Helm e `tridentctl` deployment.

Installa utilizzando l'operatore Trident

Distribuisci manualmente l'operatore Trident (modalità standard)

È possibile distribuire manualmente l'operatore Trident per installare Trident. Questa procedura si applica alle installazioni in cui le immagini dei container richieste da Trident non sono archiviate in un registro privato. Se si dispone di un registro immagini privato, utilizzare il "processo per la distribuzione offline".

Informazioni critiche su Trident 26.02

È necessario leggere le seguenti informazioni critiche su Trident.

Informazioni critiche su Trident

- Kubernetes 1.35 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  L'utilizzo di una configurazione non multipath o l'utilizzo di `find_multipaths: yes` o `find_multipaths: smart` nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'utilizzo di `find_multipaths: no` dalla release 21.07.

Distribuire manualmente l'operatore Trident e installare Trident

Verificate "la 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 e accedere prima come **system:admin** eseguendo `oc login -u system:admin` o `oc login -u kube-admin`.

1. Verifica la versione di Kubernetes:

```
kubectl version
```

2. Verificare i privilegi di amministratore del cluster:

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

3. Verifica di poter avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il tuo sistema storage tramite la rete del pod:

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

Passaggio 1: Scarica il pacchetto di installazione di Trident

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

```
wget https://github.com/NetApp/trident/releases/download/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Passaggio 2: crea la `TridentOrchestrator` CRD

Crea la `TridentOrchestrator` Definizione di Risorsa Personalizzata (CRD). Creerai una `TridentOrchestrator` Risorsa Personalizzata in seguito. Utilizza la versione YAML appropriata del CRD in `deploy/crds` per creare la `TridentOrchestrator` CRD.

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

Fase 3: distribuisci l'operatore Trident

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

- Per i cluster che eseguono Kubernetes 1.24, usa `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 distribuisce l'operatore nel `trident` namespace. Se il `trident` namespace non esiste, crearlo utilizzando:

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

- Per distribuire l'operatore in uno spazio dei nomi diverso dal `trident` namespace, aggiornare `serviceaccount.yaml`, `clusterrolebinding.yaml` e `operator.yaml` e generare il file bundle usando il `kustomization.yaml`.

  a. Crea `kustomization.yaml` utilizzando il seguente comando dove <*bundle.yaml*> è `bundle_pre_1_25.yaml` o `bundle_post_1_25.yaml` in base alla tua versione di Kubernetes.

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

  b. Compila il bundle usando il seguente comando, dove <*bundle.yaml*> è `bundle_pre_1_25.yaml` o `bundle_post_1_25.yaml` in base alla tua versione di Kubernetes.

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

Passaggi

1. Crea le risorse e distribuisci l'operatore:

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

2. Verificare che l'operatore, il deployment e i replicasets siano stati creati.

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

> Dovrebbe esserci solo **un'istanza** dell'operatore in un cluster Kubernetes. Non creare distribuzioni multiple dell'operatore Trident.

Passaggio 4: crea `TridentOrchestrator` e installa Trident

Ora puoi creare il `TridentOrchestrator` e installare Trident. Facoltativamente, puoi "personalizza la tua installazione Trident" utilizzando gli attributi nella `TridentOrchestrator` spec.

```
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:26.02
    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:26.02.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v26.02.0
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 `TridentOrchestrator` status

Lo stato di `TridentOrchestrator` indica se l'installazione è riuscita e visualizza la versione di Trident installata. Durante l'installazione, lo stato di `TridentOrchestrator` cambia da `Installing` a `Installed`. Se si osserva lo stato `Failed` e l'operatore non è in grado di ripristinarsi autonomamente, "controlla i registri".

Stato	Descrizione
Installazione	L'operatore sta installando Trident utilizzando questo `TridentOrchestrator` CR.
Installato	Trident è stato installato correttamente.
Disinstallazione	L'operatore sta disinstallando Trident, perché `spec.uninstall=true`.
Disinstallato	Trident è disinstallato.
Fallito	L'operatore non è riuscito a installare, applicare patch, aggiornare o disinstallare Trident; l'operatore tenterà automaticamente di recuperare da questo stato. Se questo stato persiste, sarà necessaria la risoluzione dei problemi.
Aggiornamento	L'operatore sta aggiornando un'installazione esistente.
Errore	The `TridentOrchestrator` non è utilizzato. Ne esiste già un altro.

Utilizzo dello stato di creazione del pod

Puoi confermare se l'installazione di Trident è stata completata esaminando 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
```

Utilizzando `tridentctl`

Puoi usare `tridentctl` per verificare la versione di Trident installata.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

Distribuisci manualmente l'operatore Trident (modalità offline)

È possibile distribuire manualmente l'operatore Trident per installare Trident. Questa procedura si applica alle installazioni in cui le immagini dei container richieste da Trident sono archiviate in un registro privato. Se non si dispone di un registro immagini privato, utilizzare il "processo per la distribuzione standard".

Informazioni critiche su Trident

È necessario leggere le seguenti informazioni critiche su Trident.

Informazioni critiche su Trident

- Kubernetes 1.35 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  L'utilizzo di una configurazione non multipath o l'utilizzo di `find_multipaths: yes` o `find_multipaths: smart` nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'utilizzo di `find_multipaths: no` dalla release 21.07.

Distribuire manualmente l'operatore Trident e installare Trident

Verificate "la 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 e accedere prima come **system:admin** eseguendo `oc login -u system:admin` o `oc login -u kube-admin`.

1. Verifica la versione di Kubernetes:

```
kubectl version
```

2. Verificare i privilegi di amministratore del cluster:

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

3. Verifica di poter avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il tuo sistema storage tramite la rete del pod:

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

Passaggio 1: Scarica il pacchetto di installazione di Trident

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

```
wget https://github.com/NetApp/trident/releases/download/v6.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Passaggio 2: crea la `TridentOrchestrator` CRD

Crea la `TridentOrchestrator` Definizione di Risorsa Personalizzata (CRD). Creerai una `TridentOrchestrator` Custom Resources più tardi. Usa la versione YAML appropriata della CRD in `deploy/crds` per creare la `TridentOrchestrator` CRD:

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

Passaggio 3: Aggiorna la posizione del registro nell'operatore

In `/deploy/operator.yaml`, aggiorna `image: docker.io/netapp/trident-operator:26.02.0` per riflettere la posizione del tuo registro delle immagini. Le tue "Trident e immagini 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:26.02.0` se tutte le immagini si trovano in un unico registro.

- `image: <your-registry>/netapp/trident-operator:26.02.0` se la tua immagine Trident si trova in un registro diverso rispetto alle tue immagini CSI.

Fase 4: distribuisci l'operatore Trident

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

- Per i cluster che eseguono Kubernetes 1.24, usa `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 distribuisce l'operatore nel `trident` namespace. Se il `trident` namespace non esiste, crearlo utilizzando:

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

- Per distribuire l'operatore in uno spazio dei nomi diverso dal `trident` namespace, aggiornare `serviceaccount.yaml`, `clusterrolebinding.yaml` e `operator.yaml` e generare il file bundle usando il `kustomization.yaml`.

  a. Crea `kustomization.yaml` utilizzando il seguente comando dove *<bundle.yaml>* è `bundle_pre_1_25.yaml` o `bundle_post_1_25.yaml` in base alla tua versione di Kubernetes.

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

  b. Compila il bundle usando il seguente comando, dove *<bundle.yaml>* è `bundle_pre_1_25.yaml` o `bundle_post_1_25.yaml` in base alla tua versione di Kubernetes.

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

Passaggi

1. Crea le risorse e distribuisci l'operatore:

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

2. Verificare che l'operatore, il deployment e i replicasets siano stati creati.

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

Dovrebbe esserci solo **un'istanza** dell'operatore in un cluster Kubernetes. Non creare distribuzioni multiple dell'operatore Trident.

Fase 5: Aggiorna la posizione del registro delle immagini nel `TridentOrchestrator`

Le tue "Trident e immagini CSI" possono essere localizzate in un unico registro o in registri diversi, ma tutte le immagini CSI devono trovarsi nello stesso registro. Aggiorna `deploy/crds/tridentorchestrator_cr.yaml` per aggiungere le specifiche di posizione aggiuntive in base alla configurazione del tuo registro.

Immagini in un registry

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

Immagini in diversi registri

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

Passaggio 6: crea `TridentOrchestrator` e installa Trident

Ora puoi creare il `TridentOrchestrator` e installare Trident. Facoltativamente, puoi "personalizza la tua installazione Trident"ulteriormente utilizzando gli attributi nella `TridentOrchestrator` specifica. Il seguente esempio mostra un'installazione in cui le immagini Trident e CSI si trovano in registri diversi.

```
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:26.02
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.02.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/trident-autosupport:26.02
    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:26.02.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v26.02.0
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 `TridentOrchestrator` status

Lo stato di `TridentOrchestrator` indica se l'installazione è riuscita e visualizza la versione di Trident installata. Durante l'installazione, lo stato di `TridentOrchestrator` cambia da `Installing` a `Installed`. Se si osserva lo stato `Failed` e l'operatore non è in grado di ripristinarsi autonomamente, "controlla i registri".

Stato	Descrizione
Installazione	L'operatore sta installando Trident utilizzando questo `TridentOrchestrator` CR.
Installato	Trident è stato installato correttamente.
Disinstallazione	L'operatore sta disinstallando Trident, perché `spec.uninstall=true`.
Disinstallato	Trident è disinstallato.
Fallito	L'operatore non è riuscito a installare, applicare patch, aggiornare o disinstallare Trident; l'operatore tenterà automaticamente di recuperare da questo stato. Se questo stato persiste, sarà necessaria la risoluzione dei problemi.
Aggiornamento	L'operatore sta aggiornando un'installazione esistente.
Errore	The `TridentOrchestrator` non è utilizzato. Ne esiste già un altro.

Utilizzo dello stato di creazione del pod

Puoi confermare se l'installazione di Trident è stata completata esaminando 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
```

Utilizzando `tridentctl`

Puoi usare `tridentctl` per verificare la versione di Trident installata.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

Distribuisci l'operatore Trident tramite Helm (modalità standard)

È possibile distribuire l'operatore Trident e installare Trident utilizzando Helm. Questa procedura si applica alle installazioni in cui le immagini dei container richieste da Trident non sono archiviate in un registro privato. Se si dispone di un registro immagini privato, utilizzare il "processo per la distribuzione offline".

Informazioni critiche su Trident 25.10

È necessario leggere le seguenti informazioni critiche su Trident.

Informazioni critiche su Trident

- Kubernetes 1.35 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  L'utilizzo di una configurazione non multipath o l'utilizzo di `find_multipaths: yes` o `find_multipaths: smart` nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'utilizzo di `find_multipaths: no` dalla release 21.07.

Distribuire l'operatore Trident e installare Trident utilizzando Helm

Utilizzando il Trident "Helm Chart" puoi distribuire l'operatore Trident e installare Trident in un unico passaggio.

Verificate "la 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 di deployment" è necessario "Helm versione 3".

Passaggi

1. Aggiungi 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..0` è la versione di Trident che stai installando.

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace <trident-namespace>
```

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

È possibile utilizzare `helm list` per esaminare i dettagli dell'installazione, come nome, namespace, chart, stato, versione dell'app 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 le sovrascritture. Questo può essere specificato più volte e il file più a destra avrà la precedenza.
`--set`	Specificare le sovrascritture sulla riga di comando.

Ad esempio, per modificare il valore predefinito di `debug`, eseguire il seguente comando dove `100.2602.0` è la versione di Trident che si sta installando:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set tridentDebug=true
```

Opzioni di configurazione

Questa tabella e il file `values.yaml`, che fa parte dell'Helm chart, forniscono l'elenco delle chiavi e i loro valori predefiniti.

Opzione	Descrizione	Predefinito
`nodeSelector`	Etichette dei nodi per l'assegnazione dei pod	
`podAnnotations`	Annotazioni del pod	

Opzione	Descrizione	Predefinito
`deploymentAnnotations`	Annotazioni di distribuzione	
`tolerations`	Tolleranze per l'assegnazione dei pod	

`affinity`	Affinità per l'assegnazione dei 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. Quando vuoi fornire un'affinità personalizzata, estendi l'affinità predefinita.

`tridentControllerPluginNodeSelector`	Selettori di nodo aggiuntivi per i pod. Consultare Comprendere i pod controller e i pod nodo per i dettagli.	
`tridentControllerPluginTolerations`	Sostituisce le tolleranze di Kubernetes per i pod. Consultare Comprendere i pod controller e i pod nodo per i dettagli.	
`tridentNodePluginNodeSelector`	Selettori di nodo aggiuntivi per i pod. Consultare Comprendere i pod controller e i pod nodo per i dettagli.	

Opzione	Descrizione	Predefinito
`tridentNodePluginTolerations`	Sostituisce le tolleranze di Kubernetes per i pod. Consultare Comprendere i pod controller e i pod nodo per i dettagli.	
`imageRegistry`	Identifica il registro per le `trident-operator`, `trident` e altre immagini. Lascia vuoto per accettare l'impostazione predefinita. IMPORTANTE: Quando si installa Trident in un repository privato, se si utilizza l'opzione `imageRegistry` per specificare la posizione del repository, non usare `/netapp/` nel percorso del repository.	`""`
`imagePullPolicy`	Imposta il criterio di pull dell'immagine per il `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Imposta i segreti di estrazione delle immagini per le `trident-operator`, `trident` e altre immagini.	
`kubeletDir`	Consente di sovrascrivere la posizione host dello stato interno di 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 la completa sostituzione dell'immagine per `trident-operator`.	`""`
`operatorImageTag`	Permette di sovrascrivere il tag dell' `trident-operator` immagine.	`""`
`tridentIPv6`	Consente di abilitare Trident per funzionare in cluster IPv6.	`false`
`tridentK8sTimeout`	Sostituisce il timeout predefinito di 30 secondi per la maggior parte delle operazioni API di Kubernetes (se diverso da zero, in secondi).	`0`

Opzione	Descrizione	Predefinito
`tridentHttpRequestTimeout`	Sovrascrive il timeout predefinito di 90 secondi per le richieste HTTP, con `0s` che rappresenta una durata infinita per il timeout. Non sono ammessi valori negativi.	`"90s"`
`tridentSilenceAutosupport`	Consente di disabilitare la segnalazione periodica AutoSupport di Trident.	`false`
`tridentAutosupportImageTag`	Consente di sovrascrivere il tag dell'immagine per il container Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Consente al container Trident AutoSupport di telefonare a casa tramite un proxy HTTP.	`""`
`tridentLogFormat`	Imposta il formato di logging Trident (`text` o `json`).	`"text"`
`tridentDisableAuditLog`	Disattiva il logger di audit di Trident.	`true`
`tridentLogLevel`	Consente di impostare il livello di log di Trident su: `trace`, `debug`, `info`, `warn`, `error`, o `fatal`.	`"info"`
`tridentDebug`	Consente di impostare il livello di log di Trident su `debug`. È possibile automatizzare il processo di distacco forzato tramite integrazione con node health check (NHC) operator. Per informazioni, vedi "Automatizzare il failover delle applicazioni stateful con Trident".	`false`
`tridentLogWorkflows`	Consente di abilitare flussi di lavoro Trident specifici per la registrazione delle tracce o la soppressione dei log.	`""`
`tridentLogLayers`	Consente di abilitare specifici livelli Trident per la registrazione delle tracce o la soppressione dei log.	`""`
`tridentImage`	Consente la completa sostituzione dell'immagine per Trident.	`""`
`tridentImageTag`	Consente di sovrascrivere il tag dell'immagine per Trident.	`""`
`tridentProbePort`	Consente di sovrascrivere la porta predefinita utilizzata per le sonde liveness/readiness di Kubernetes.	`""`

Opzione	Descrizione	Predefinito
`windows`	Consente l'installazione di Trident sul nodo worker di Windows.	`false`
`enableForceDetach`	Consente di abilitare la funzione force detach.	`false`
`excludePodSecurityPolicy`	Esclude la pod security policy dell'operatore dalla creazione.	`false`
`cloudProvider`	Impostare su `"Azure"` quando si utilizzano identità gestite o un'identità cloud su un cluster AKS. Impostare su "AWS" quando si utilizza un'identità cloud su un cluster EKS.	`""`
`cloudIdentity`	Impostare su workload identity ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx") quando si utilizza l'identità cloud su un cluster AKS. Impostare su AWS IAM role ("'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role'") quando si utilizza l'identità cloud su un cluster EKS.	`""`
`iscsiSelfHealingInterval`	L'intervallo al quale viene invocata la funzione di auto-riparazione iSCSI.	`5m0s`
`iscsiSelfHealingWaitTime`	La durata dopo la quale la funzione di auto-riparazione iSCSI avvia un tentativo di risolvere una sessione non aggiornata eseguendo un logout e un successivo login.	`7m0s`
`nodePrep`	Consente 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.** NOTA: a partire da OpenShift 4.19, la versione minima di Trident supportata per questa funzionalità è la 25.06.1.	

Opzione	Descrizione	Predefinito
`enableConcurrency`	Consente operazioni simultanee del controller Trident per un throughput migliorato. **Anteprima tecnica**: questa funzionalità è sperimentale e attualmente supporta flussi di lavoro paralleli limitati con i driver ONTAP-NAS (solo NFS) e ONTAP-SAN (NVMe per unified ONTAP 9), oltre all'anteprima tecnica esistente per il driver ONTAP-SAN (protocolli iSCSI e FCP in unified ONTAP 9).	falso
`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. Puoi configurare CPU e memoria per ogni container 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 container o campo.
	• NON modificare l'indentazione - l'indentazione YAML è fondamentale per una corretta analisi.
		<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:
		    linux:
		      trident-main:</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 e backend in una rete multi-home.	falso

Comprendere i pod controller e i pod nodo

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

Kubernetes "selettori di nodi" e "tolleranze e taint" sono utilizzati per vincolare un pod a essere eseguito su un nodo specifico o preferito. Utilizzando `ControllerPlugin` e `NodePlugin`, puoi specificare vincoli e override.

- Il plugin del controller gestisce il provisioning e la gestione dei volumi, come snapshot e ridimensionamento.
- Il plugin del nodo gestisce il collegamento dello storage al nodo.

Distribuire l'operatore Trident utilizzando Helm (Offline mode)

È possibile distribuire l'operatore Trident e installare Trident utilizzando Helm. Questa procedura si applica alle installazioni in cui le immagini dei container richieste da Trident sono archiviate in un registro privato. Se non si dispone di un registro immagini privato, utilizzare il "processo per la distribuzione standard".

Informazioni critiche su Trident 26.02

È necessario leggere le seguenti informazioni critiche su Trident.

Informazioni critiche su Trident

- Kubernetes 1.35 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  L'utilizzo di una configurazione non multipath o l'utilizzo di `find_multipaths: yes` o `find_multipaths: smart` nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'utilizzo di `find_multipaths: no` dalla release 21.07.

Distribuire l'operatore Trident e installare Trident utilizzando Helm

Utilizzando il Trident "Helm Chart" puoi distribuire l'operatore Trident e installare Trident in un unico passaggio.

Verificate "la 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 di deployment" è necessario "Helm versione 3".

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

Passaggi

1. Aggiungi 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 "Trident e immagini CSI" possono essere localizzate in un unico registro o in registri diversi, ma tutte le immagini CSI devono trovarsi nello stesso registro. Negli esempi, `100.2602.0` è la versione di Trident che stai installando.

Immagini in un registry

```
helm install <name> netapp-trident/trident-operator --version
100.2602.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace> --set nodePrep={iscsi}
```

Immagini in diversi registri

```
helm install <name> netapp-trident/trident-operator --version
100.2602.0 --set imageRegistry=<your-registry> --set operatorImage
=<your-registry>/trident-operator:26.02.0 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:26.02
--set tridentImage=<your-registry>/trident:26.02.0 --create
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```

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

È possibile utilizzare `helm list` per esaminare i dettagli dell'installazione, come nome, namespace, chart, stato, versione dell'app 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 le sovrascritture. Questo può essere specificato più volte e il file più a destra avrà la precedenza.
`--set`	Specificare le sovrascritture sulla riga di comando.

Ad esempio, per modificare il valore predefinito di `debug`, eseguire il seguente comando dove `100.2602.0` è la versione di Trident che si sta installando:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set tridentDebug=true
```

Per aggiungere il valore nodePrep, eseguire il seguente comando:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set nodePrep={iscsi}
```

Opzioni di configurazione

Questa tabella e il file `values.yaml`, che fa parte dell'Helm chart, forniscono l'elenco delle chiavi e i loro valori predefiniti.

> Non rimuovere l'affinità predefinita dal file values.yaml. Quando vuoi fornire un'affinità personalizzata, estendi l'affinità predefinita.

Opzione	Descrizione	Predefinito
`nodeSelector`	Etichette dei nodi per l'assegnazione dei pod	
`podAnnotations`	Annotazioni del pod	
`deploymentAnnotations`	Annotazioni di distribuzione	
`tolerations`	Tolleranze per l'assegnazione dei pod	

Opzione	Descrizione	Predefinito

`affinity`	Affinità per l'assegnazione dei 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. Quando vuoi fornire un'affinità personalizzata, estendi l'affinità predefinita.

`tridentControllerPluginNodeSelector`	Selettori di nodo aggiuntivi per i pod. Consultare "Comprendere i pod controller e i pod nodo" per i dettagli.	
`tridentControllerPluginTolerations`	Sostituisce le tolleranze di Kubernetes per i pod. Consultare "Comprendere i pod controller e i pod nodo" per i dettagli.	

Opzione	Descrizione	Predefinito
`tridentNodePluginNodeSelector`	Selettori di nodo aggiuntivi per i pod. Consultare "Comprendere i pod controller e i pod nodo" per i dettagli.	
`tridentNodePluginTolerations`	Sostituisce le tolleranze di Kubernetes per i pod. Consultare "Comprendere i pod controller e i pod nodo" per i dettagli.	
`imageRegistry`	Identifica il registro per le `trident-operator`, `trident` e altre immagini. Lascia vuoto per accettare l'impostazione predefinita. IMPORTANTE: Quando si installa Trident in un repository privato, se si utilizza l'opzione `imageRegistry` per specificare la posizione del repository, non usare `/netapp/` nel percorso del repository.	""
`imagePullPolicy`	Imposta il criterio di pull dell'immagine per il `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Imposta i segreti di estrazione delle immagini per le `trident-operator`, `trident` e altre immagini.	
`kubeletDir`	Consente di sovrascrivere la posizione host dello stato interno di 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 la completa sostituzione dell'immagine per `trident-operator`.	""
`operatorImageTag`	Permette di sovrascrivere il tag dell' `trident-operator` immagine.	""
`tridentIPv6`	Consente di abilitare Trident per funzionare 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` parametro è applicabile solo per l'installazione Trident.	`180`
`tridentHttpRequestTimeout`	Sovrascrive il timeout predefinito di 90 secondi per le richieste HTTP, con `0s` che rappresenta una durata infinita per il timeout. Non sono ammessi valori negativi.	`"90s"`
`tridentSilenceAutosupport`	Consente di disabilitare la segnalazione periodica AutoSupport di Trident.	`false`
`tridentAutosupportImageTag`	Consente di sovrascrivere il tag dell'immagine per il container Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Consente al container Trident AutoSupport di telefonare a casa tramite un proxy HTTP.	""
`tridentLogFormat`	Imposta il formato di logging Trident (`text` o `json`).	`"text"`
`tridentDisableAuditLog`	Disattiva il logger di audit di Trident.	`true`
`tridentLogLevel`	Consente di impostare il livello di log di Trident su: `trace`, `debug`, `info`, `warn`, `error`, o `fatal`.	`"info"`
`tridentDebug`	Consente di impostare il livello di log di Trident su `debug`.	`false`
`tridentLogWorkflows`	Consente di abilitare flussi di lavoro Trident specifici per la registrazione delle tracce o la soppressione dei log.	""
`tridentLogLayers`	Consente di abilitare specifici livelli Trident per la registrazione delle tracce o la soppressione dei log.	""
`tridentImage`	Consente la completa sostituzione dell'immagine per Trident.	""
`tridentImageTag`	Consente di sovrascrivere il tag dell'immagine per Trident.	""

Opzione	Descrizione	Predefinito
`tridentProbePort`	Consente di sovrascrivere la porta predefinita utilizzata per le sonde liveness/readiness di Kubernetes.	""
`windows`	Consente l'installazione di Trident sul nodo worker di Windows.	`false`
`enableForceDetach`	Consente di abilitare la funzione force detach. È possibile automatizzare il processo di distacco forzato tramite integrazione con node health check (NHC) operator. Per informazioni, vedi "Automatizzare il failover delle applicazioni stateful con Trident".	`false`
`excludePodSecurityPolicy`	Esclude la pod security policy dell'operatore dalla creazione.	`false`
`nodePrep`	Consente 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 di Trident supportata per questa funzionalità è la 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. Puoi configurare CPU e memoria per ogni container 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 container o campo.
	• NON modificare l'indentazione - l'indentazione YAML è fondamentale per una corretta analisi.
		`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:`
		`  linux:`

Personalizzare l'installazione dell'operatore Trident

L'operatore Trident permette di personalizzare l'installazione Trident usando gli attributi nella `TridentOrchestrator` spec. Se vuoi personalizzare l'installazione oltre ciò che gli argomenti di `TridentOrchestrator` consentono, considera l'utilizzo di `tridentctl` per generare manifesti YAML personalizzati da modificare secondo necessità.

Comprendere i pod controller e i pod nodo

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

Kubernetes "selettori di nodi" e "tolleranze e taint" sono utilizzati per vincolare un pod a essere eseguito su un nodo specifico o preferito. Utilizzando `ControllerPlugin` e `NodePlugin`, puoi specificare vincoli e override.

- Il plugin del controller gestisce il provisioning e la gestione dei volumi, come snapshot e ridimensionamento.
- Il plugin del nodo gestisce il collegamento dello storage al nodo.

Opzioni di configurazione

> **!** `spec.namespace` è specificato in `TridentOrchestrator` per indicare il namespace in cui Trident è installato. Questo parametro **non può essere aggiornato dopo che Trident è installato**. Il tentativo di farlo fa sì che lo stato `TridentOrchestrator` cambi in `Failed`. Trident non è destinato a essere migrato tra namespace.

Questa tabella illustra `TridentOrchestrator` gli attributi.

Parametro	Descrizione	Predefinito
`namespace`	Spazio dei nomi in cui installare Trident	`"default"`
`debug`	Abilita il debug per Trident	`false`
`enableForceDetach`	`ontap-san`, `ontap-san-economy`, `ontap-nas` e `ontap-nas-economy` solo. Funziona con Kubernetes Non-Graceful Node Shutdown (NGNS) per garantire agli amministratori cluster la possibilità di migrare in modo sicuro i carichi di lavoro con volumi montati su nuovi nodi nel caso in cui un nodo diventi non sano. Per informazioni, vedi "Automatizzare il failover delle applicazioni stateful con Trident".	`false`
`windows`	L'impostazione su `true` abilita l'installazione sui nodi worker Windows.	`false`

Parametro	Descrizione	Predefinito
`cloudProvider`	Impostare su `"Azure"` quando si utilizzano identità gestite o un'identità cloud su un cluster AKS. Impostare su `"AWS"` quando si utilizza una cloud identity su un cluster EKS. Impostare su `"GCP"` quando si utilizza una cloud identity su un cluster GKE.	`""`
`cloudIdentity`	Impostare su workload identity ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx") quando si utilizza l'identità cloud su un cluster AKS. Impostare su AWS IAM role ("'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role'") quando si utilizza l'identità cloud su un cluster EKS. Impostare su cloud identity ("'iam.gke.io/gcp-service-account: xxxx@mygcpproject.iam.gserviceaccount.com'") quando si utilizza cloud identity su un cluster GKE.	`""`
`IPv6`	Install Trident su IPv6	falso
`k8sTimeout`	Timeout per le operazioni Kubernetes. Il `k8sTimeout` parametro è applicabile solo per l'installazione Trident.	`180sec`
`silenceAutosupport`	Non inviare automaticamente i bundle di autosupport a NetApp	`false`
`autosupportImage`	L'immagine container per Autosupport Telemetry	`"netapp/trident-autosupport10"`
`autosupportProxy`	L'indirizzo/porta di un proxy per l'invio della Telemetria di Autosupport	`"http://proxy.example.com:8888"`
`uninstall`	Un flag utilizzato per disinstallare Trident	`false`
`logFormat`	Formato di registrazione Trident da utilizzare [text,json]	`"text"`
`tridentImage`	Trident image da installare	`"netapp/trident:26.02"`
`imageRegistry`	Percorso al registro interno, del formato `<registry FQDN>[:port][/subpath]`	`"registry.k8s.io"`
`kubeletDir`	Percorso della directory kubelet sull'host	`"/var/lib/kubelet"`
`wipeout`	Un elenco di risorse da eliminare per eseguire una rimozione completa di Trident	
`imagePullSecrets`	Segreti per prelevare immagini da un registro interno	

Parametro	Descrizione	Predefinito
`imagePullPolicy`	Imposta la policy di pull dell'immagine per il Trident operator. I valori validi sono: `Always` per estrarre sempre l'immagine. `IfNotPresent` per estrarre l'immagine solo se non esiste già sul nodo. `Never` per non estrarre mai l'immagine.	`IfNotPresent`
`controllerPluginNodeSelector`	Selettori di nodo aggiuntivi per i pod. Segue lo stesso formato come `pod.spec.nodeSelector`.	Nessun valore predefinito; facoltativo
`controllerPluginTolerations`	Sostituisce le tolleranze di Kubernetes per i pod. Segue lo stesso formato di `pod.spec.Tolerations`.	Nessun valore predefinito; facoltativo
`nodePluginNodeSelector`	Selettori di nodo aggiuntivi per i pod. Segue lo stesso formato come `pod.spec.nodeSelector`.	Nessun valore predefinito; facoltativo
`nodePluginTolerations`	Sostituisce le tolleranze di Kubernetes per i pod. Segue lo stesso formato di `pod.spec.Tolerations`.	Nessun valore predefinito; facoltativo
`nodePrep`	Consente 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 di Trident supportata per questa funzionalità è la 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 migliorato. **Anteprima tecnica**: questa funzionalità è sperimentale e attualmente supporta flussi di lavoro paralleli limitati con i driver ONTAP-NAS (solo NFS) e ONTAP-SAN (NVMe per unified ONTAP 9), oltre all'anteprima tecnica esistente per il driver ONTAP-SAN (protocolli iSCSI e FCP in unified ONTAP 9).	falso

Parametro	Descrizione	Predefinito
`resources`	Imposta i limiti delle risorse Kubernetes e le richieste per il controller Trident e i pod dei nodi. Puoi configurare CPU e memoria per ogni container 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 container 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 sono applicate automaticamente se non specificate. • I nomi dei container sono elencati così come appaiono nelle specifiche del pod. • I sidecar sono elencati sotto ogni container principale. • Controllare il campo `status.CurrentInstallationParams` TORC per visualizzare i valori attualmente applicati.	`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:`

Parametro	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 e backend in una rete multi-home.	falso

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

Configurazioni di esempio

È possibile utilizzare gli attributi in Opzioni di configurazione quando si definisce `TridentOrchestrator` per personalizzare l'installazione.

Configurazione personalizzata di base

Questo esempio, creato dopo l'esecuzione del comando `cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml`, 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 node selectors.

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

memory:
liveness-

Nodi worker 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

Questo esempio installa 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

Questo esempio installa Trident per l'uso 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

Questo esempio installa Trident per l'uso 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

Questo esempio installa Trident per l'uso 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 di risorse Kubernetes e dei limiti per il controller Trident e i pod dei nodi Linux Trident

Questo esempio configura le richieste e i limiti di risorse Kubernetes per il controller Trident e i pod Trident del nodo Linux.

Disclaimer: I valori di richiesta e limite forniti in questo esempio sono solo a scopo dimostrativo. Adatta questi valori in base al tuo 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 di risorse Kubernetes e dei limiti per il controller Trident e per i pod dei nodi Trident Windows e Linux

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

Disclaimer: I valori di richiesta e limite forniti in questo esempio sono solo a scopo dimostrativo. Adatta questi valori in base al tuo 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 utilizzando tridentctl

Installare utilizzando tridentctl

È possibile installare Trident usando `tridentctl`. Questa procedura si applica alle installazioni in cui le immagini dei container richieste da Trident sono archiviate sia in un registro privato che no. Per personalizzare la tua `tridentctl` distribuzione, fare riferimento a "Personalizza la distribuzione di tridentctl".

Informazioni critiche su Trident10

È necessario leggere le seguenti informazioni critiche su Trident.

Informazioni critiche su Trident

- Kubernetes 1.27 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  L'utilizzo di una configurazione non multipath o l'utilizzo di `find_multipaths: yes` o `find_multipaths: smart` nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'utilizzo di `find_multipaths: no` dalla release 21.07.

Installa Trident usando `tridentctl`

Verificate "la 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 e accedere prima come **system:admin** eseguendo `oc login -u system:admin` o `oc login -u kube-admin`.

1. Verifica la versione di Kubernetes:

```
kubectl version
```

2. Verificare i privilegi di amministratore del cluster:

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

3. Verifica di poter avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il tuo sistema storage tramite la rete del pod:

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

Passaggio 1: Scarica il pacchetto di installazione di Trident

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

```
wget https://github.com/NetApp/trident/releases/download/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Passaggio 2: installa Trident

Installa Trident nello spazio dei nomi desiderato eseguendo il `tridentctl install` comando. Puoi aggiungere argomenti aggiuntivi per specificare la posizione del registro delle immagini.

Modalità standard

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

Immagini in un registry

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.02 --trident
-image <your-registry>/trident:26.02.0
```

Immagini in diversi registri

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.02 --trident
-image <your-registry>/trident:26.02.0
```

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=26.10.0
INFO Trident installation succeeded.
....
```

Verificare l'installazione

Puoi verificare la tua installazione utilizzando lo stato di creazione del pod o `tridentctl`.

Utilizzo dello stato di creazione del pod

Puoi confermare se l'installazione di Trident è stata completata esaminando 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 viene completato correttamente o `trident-controller-<generated id>` (`trident-csi-<generated id>` nelle versioni precedenti alla 23.01) non presenta lo stato **Running**, la piattaforma non è stata installata. Usa `-d` per "attiva la modalità debug" e risolvere il problema.

Utilizzando `tridentctl`

Puoi usare `tridentctl` per verificare la versione di Trident installata.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

Configurazioni di esempio

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

Nodi Windows

Per abilitare Trident sui nodi Windows:

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

Forza distacco

Per informazioni, vedi "Automatizzare il 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 per migliorare il throughput, aggiungere l'opzione `--enable-concurrency` durante l'installazione come mostrato in questo esempio.

> **Anteprima tecnica**: questa funzionalità è sperimentale e attualmente supporta flussi di lavoro paralleli limitati con i driver ONTAP-NAS (solo NFS) e ONTAP-SAN (NVMe per unified ONTAP 9), oltre all'anteprima tecnica esistente per il driver ONTAP-SAN (protocolli iSCSI e FCP in unified ONTAP 9).

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

Personalizza l'installazione di tridentctl

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

Informazioni sul programma di installazione

Il programma di installazione Trident consente di personalizzare gli attributi. Ad esempio, se hai copiato l'immagine Trident in un repository privato, puoi specificare il nome dell'immagine usando `--trident-image`. Se hai copiato l'immagine Trident e anche le immagini CSI sidecar necessarie in un repository privato, potrebbe essere preferibile specificare la posizione di quel repository usando lo switch `--image-registry`, che assume la forma `<registry FQDN>[:port]`.

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

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

Se si desidera personalizzare l'installazione al di là di quanto consentito dagli argomenti del programma di installazione, è possibile personalizzare anche i file di deployment. L'utilizzo del parametro `--generate-custom-yaml` crea i seguenti file YAML nella directory del programma di installazione `setup`:

- `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, puoi modificarli secondo le tue esigenze e poi usare `--use-custom-yaml` per installare la tua distribuzione personalizzata.

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

Installa utilizzando l'operatore certificato OpenShift

Installare Trident utilizzando OpenShift OperatorHub

Se si utilizza Red Hat OpenShift, è possibile installare NetApp Trident utilizzando l'operatore certificato Red Hat. Utilizzare questa procedura per installare Trident dalla piattaforma Red Hat OpenShift Container Platform.

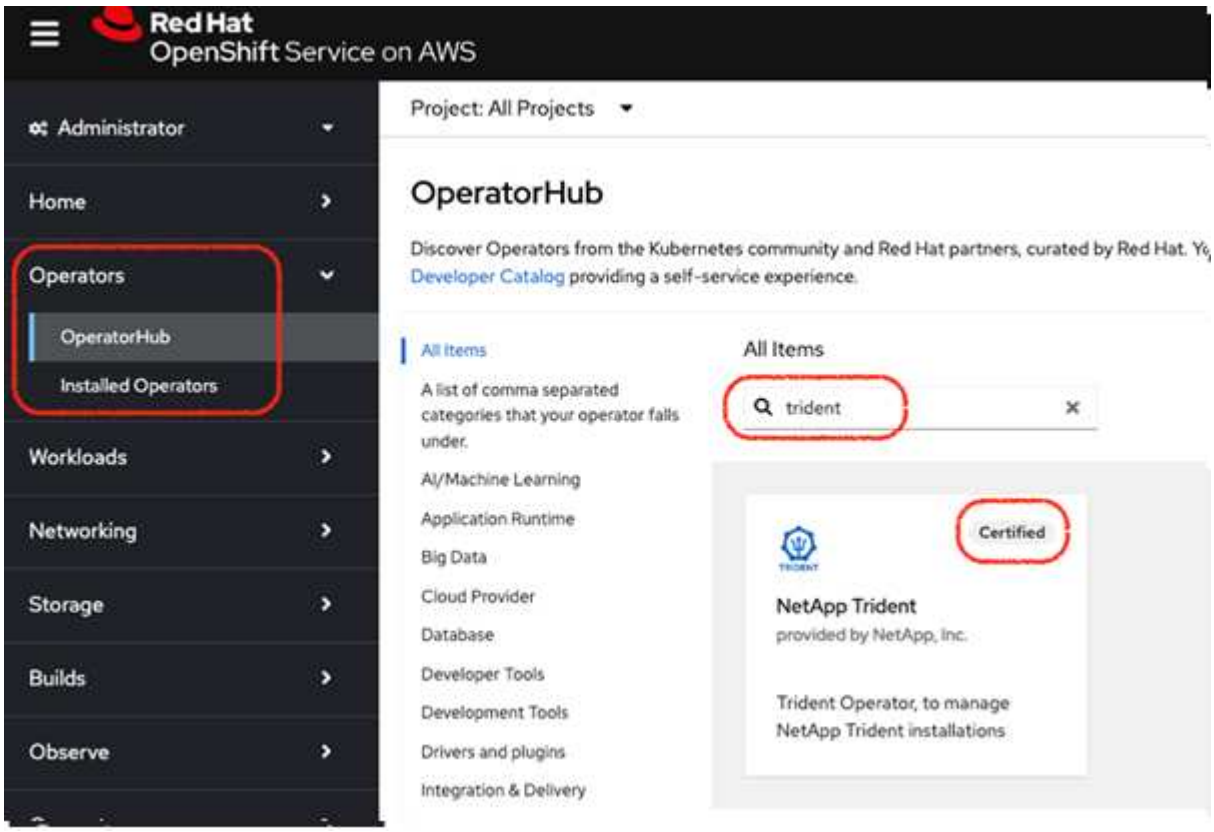
Prima di iniziare

Prima di iniziare l'installazione, "prepara il tuo ambiente per l'installazione di Trident".

Trova e installa l'operatore Trident

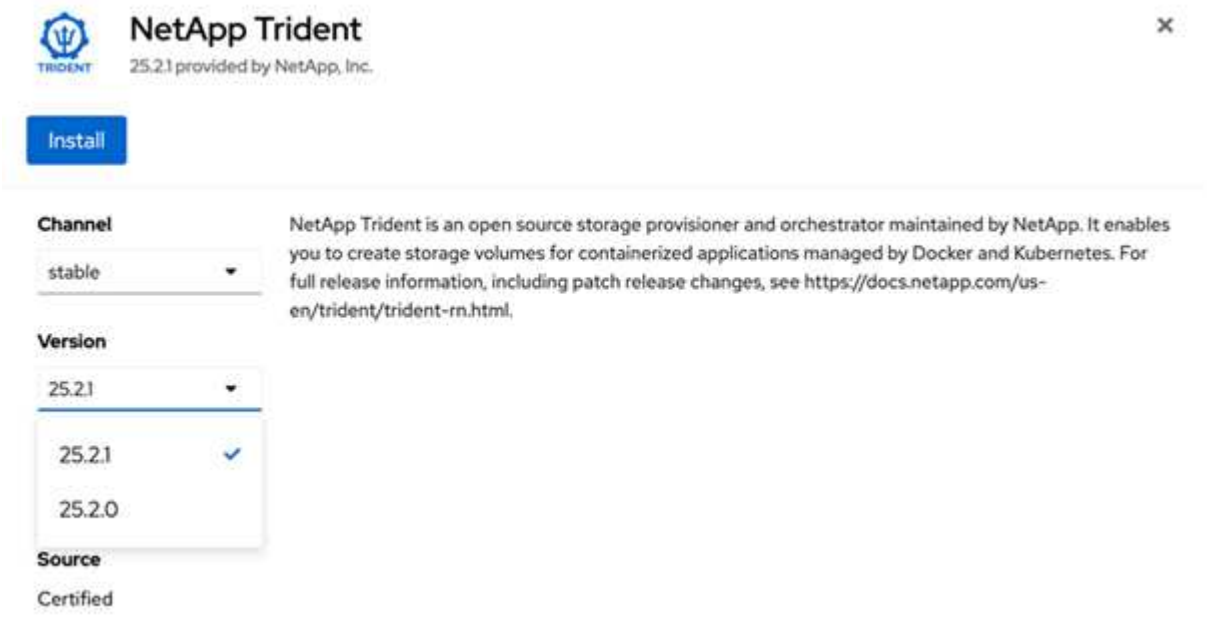
Passaggi

1. Naviga su OpenShift OperatorHub e cerca NetApp Trident.



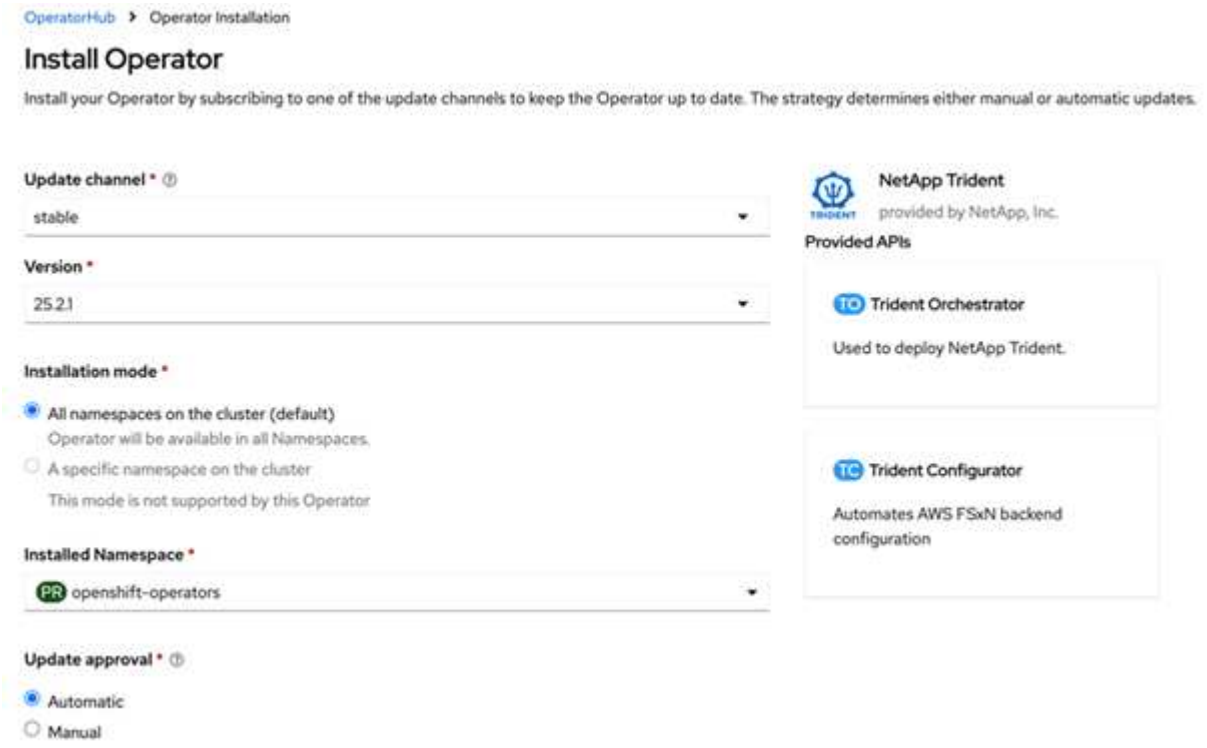
2. Fare clic su **NetApp Trident** per aprire le impostazioni di installazione.

3. Seleziona le opzioni richieste e fai clic su **Install** per aprire la configurazione dell'Operator.

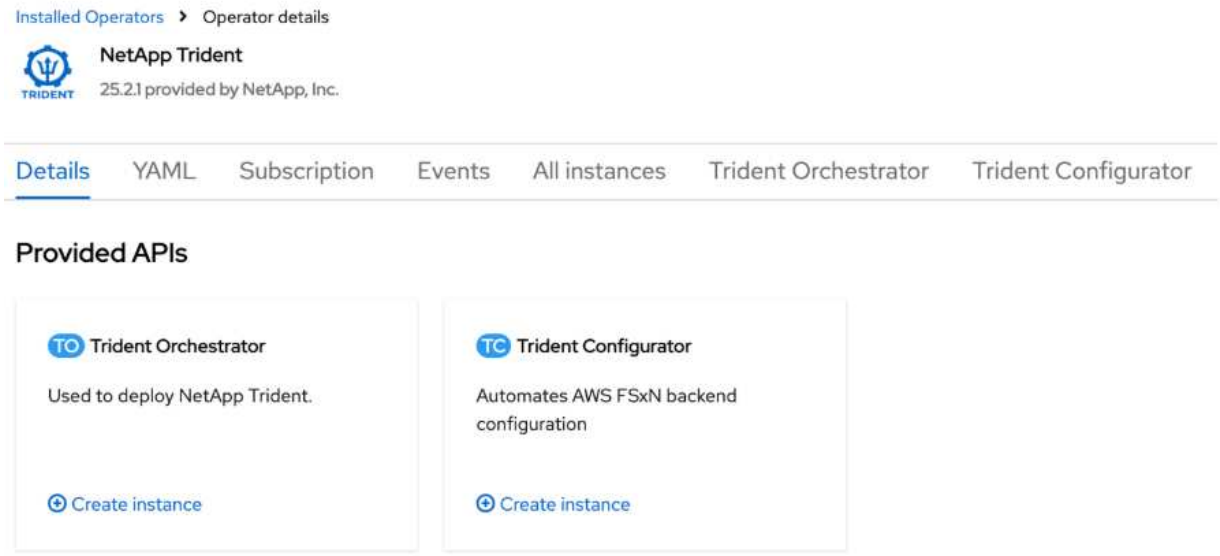


Assicurati di selezionare la versione più recente di Operator.

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



5. Fare clic su **View Operator** per visualizzare i dettagli dell'Operator.



6. Fare clic su **Vista 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.

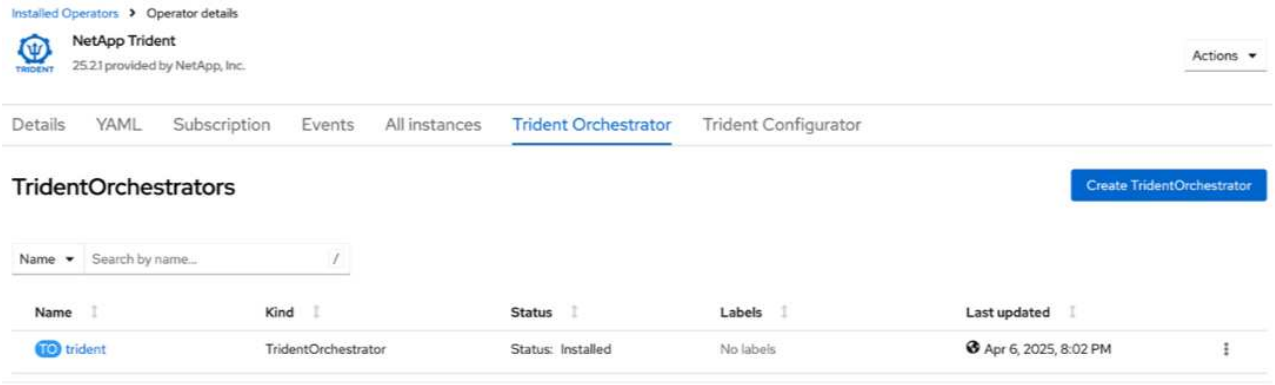
Opzionale: Abilita la concorrenza

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

```
enableConcurrency: true
```

- Red Hat Enterprise Linux CoreOS (RHCOS) non ha iSCSI abilitato e configurato.
- È possibile aggiungere il `nodePrep` parametro per configurare e abilitare entrambi i servizi iSCSI e Multipath su tutti i nodi worker OpenShift.
- A partire da OpenShift 4.19, la versione minima di Trident supportata per questa funzionalità è la 25.06.1.

1. Fare clic su **Create**; il Trident Orchestrator sarà completamente installato.



Disinstallare l'operatore Trident

Passaggi

1. Seleziona l'operatore Trident dall'elenco degli operator installati.
2. Seleziona se vuoi 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

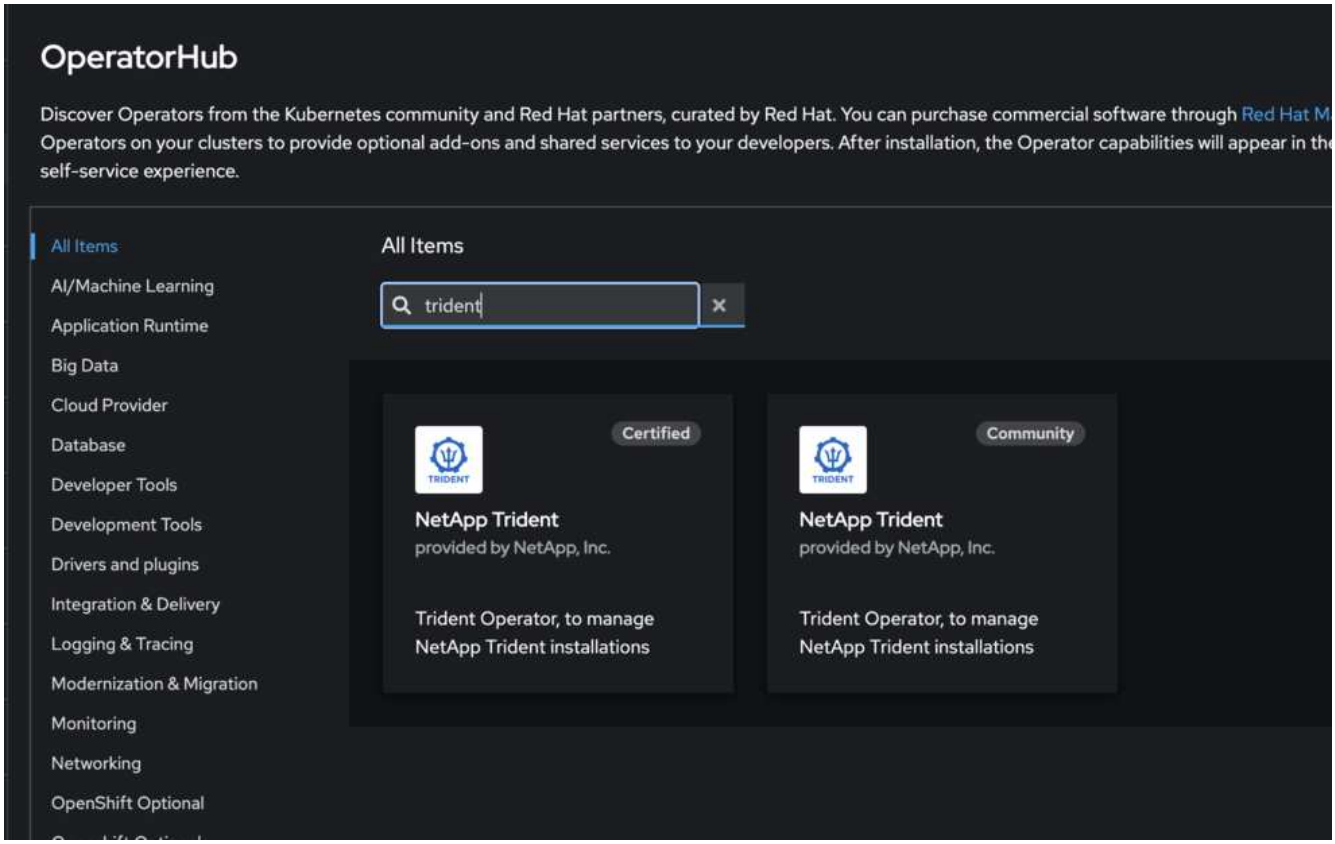
Prima di iniziare l'installazione, "prepara il tuo ambiente per l'installazione di Trident".

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.

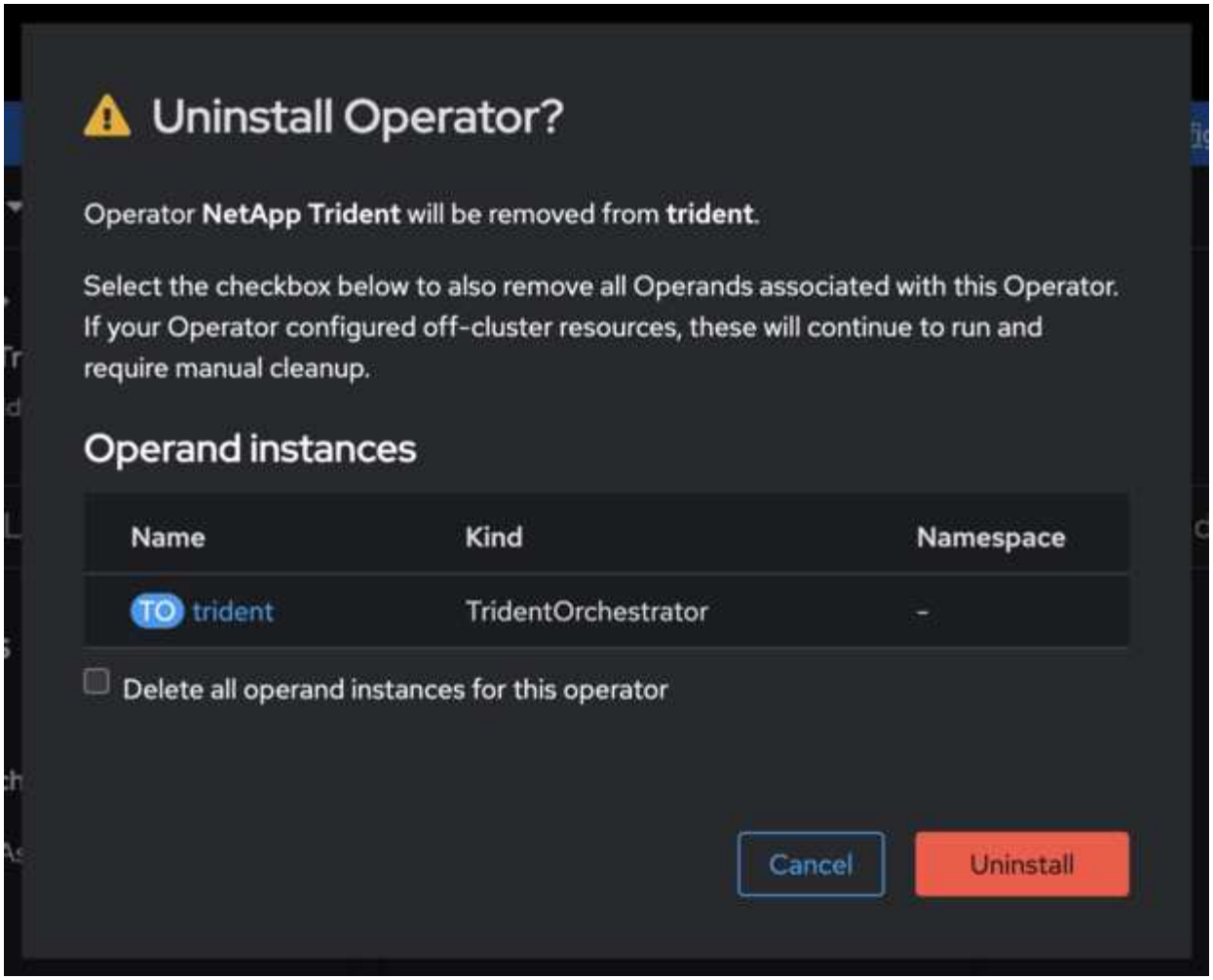
Passa dall'operatore della community

Passaggi

1. Utilizzare la console OpenShift per navigare verso la OperatorHub.



2. Trova l'operatore della comunità 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

Passaggi

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.

Passaggi

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

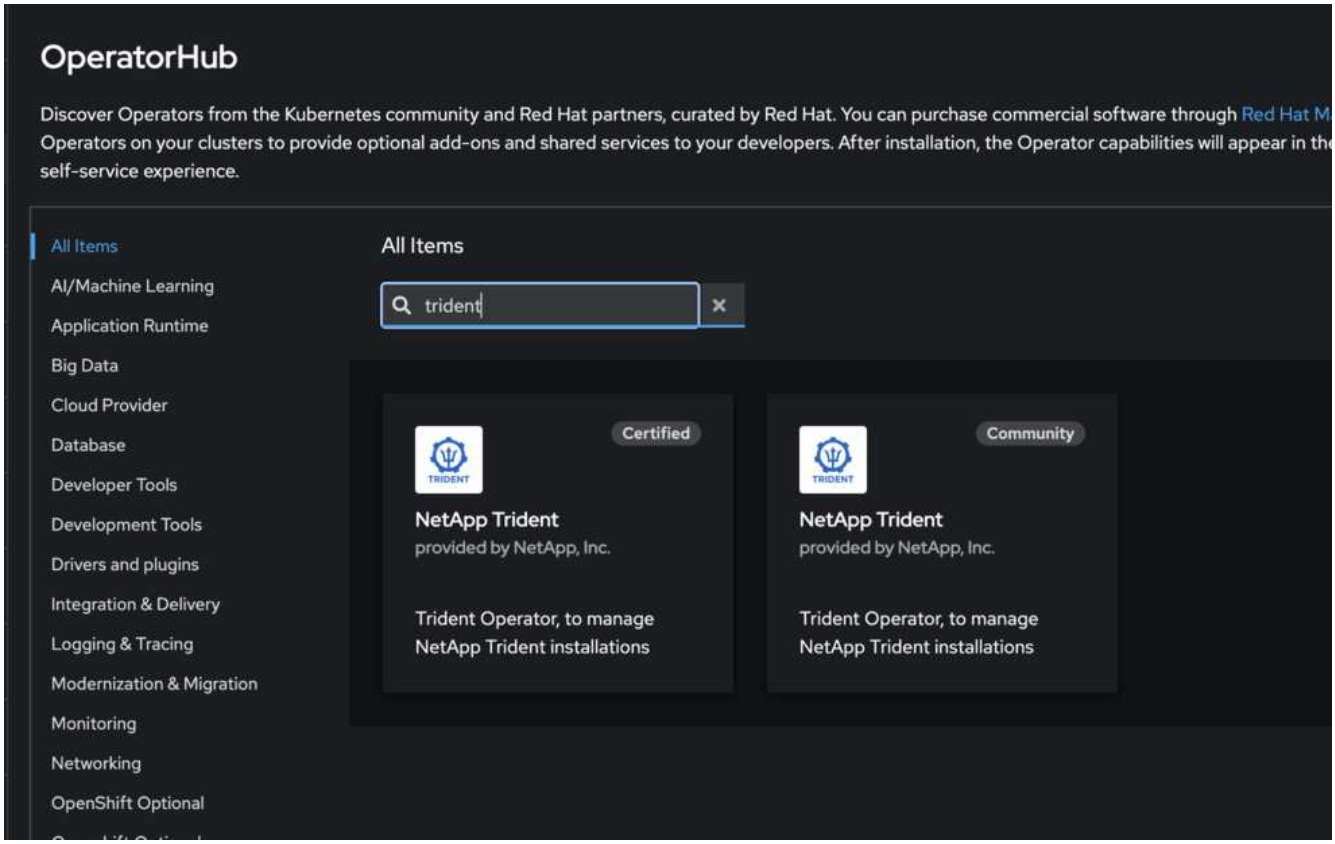
```
kubectl delete -f deploy/bundle.yaml -n trident
```

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

Installa l'operatore certificato OpenShift

Passaggi

1. Accedere a Red Hat OperatorHub.
2. Cerca e seleziona l'operatore NetApp Trident.



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

Verifica

- Controlla OperatorHub nella console per assicurarti che il nuovo operatore certificato sia stato installato correttamente.

Usa Trident

Prepara il nodo worker

Tutti i nodi worker nel cluster Kubernetes devono essere in grado di montare i volumi che hai fornito per i tuoi pod. Per preparare i nodi worker, 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 di default.

Strumenti NFS

"Installa 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 stai utilizzando: `ontap-san`, `ontap-san-economy`, `solidfire-san`.

Strumenti NVMe

"Installa gli strumenti NVMe" se si utilizza `ontap-san` per nonvolatile memory express (NVMe) su TCP (NVMe/TCP) protocollo.

> NetApp consiglia ONTAP 9.12 o versione successiva per NVMe/TCP.

Strumenti SCSI su FC

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

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

Punti da considerare: * SCSI su FC è supportato su OpenShift e KubeVirt ambienti. * SCSI su FC non è supportato su Docker. * L'auto-riparazione iSCSI non è applicabile a SCSI su FC.

Strumenti SMB

"Prepararsi al provisioning dei volumi SMB" se si utilizza: `ontap-nas` per fornire volumi SMB.

Rilevamento del servizio nodo

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

> La rilevazione dei servizi del nodo identifica i servizi rilevati, ma non garantisce che siano configurati correttamente. Al contrario, l'assenza di un servizio rilevato non garantisce che il montaggio del volume fallirà.

Rivedi gli eventi

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

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

Esamina i servizi scoperti

Trident identifica i servizi abilitati per ciascun nodo sul Trident node 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. Assicurati che il servizio NFS sia avviato durante il tempo di avvio.

RHEL 8+

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

Ubuntu

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

Riavvia i nodi worker dopo aver installato gli strumenti NFS per prevenire errori durante il collegamento dei volumi ai container.

volumi iSCSI

Trident può stabilire automaticamente una sessione iSCSI, analizzare le LUN, rilevare dispositivi multipath, formattarli e montarli su un pod.

Capacità di auto-riparazione iSCSI

Per i sistemi ONTAP, Trident esegue l'auto-riparazione iSCSI ogni cinque minuti per:

1. **Identifica** lo stato della sessione iSCSI desiderato e lo stato della sessione iSCSI corrente.
2. **Confronta** lo stato desiderato con quello attuale per identificare le riparazioni necessarie. Trident determina le priorità di riparazione e quando preemptare le riparazioni.
3. **Eseguire le riparazioni** necessarie per riportare lo stato della sessione iSCSI corrente allo stato della sessione iSCSI desiderato.

I log delle attività di auto-riparazione si trovano nel `trident-main` container sul rispettivo pod Daemonset. Per visualizzare i log, è necessario aver impostato `debug` su "true" durante l'installazione di Trident.

Le funzionalità di auto-riparazione iSCSI di Trident possono aiutare a prevenire:

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

> Ad esempio, se i segreti CHAP vengono ruotati sul storage controller e la rete perde la connettività, i vecchi (*obsoleti*) segreti CHAP potrebbero persistere. La funzione di auto-riparazione può riconoscere questo e ristabilire automaticamente la sessione per applicare i segreti CHAP aggiornati.

- Sessioni iSCSI mancanti
- LUN mancanti

Punti da considerare prima di aggiornare Trident

- Se vengono utilizzati solo igroup per nodo (introdotti nella versione 23.04+), la funzione di auto-riparazione iSCSI avvierà nuove scansioni SCSI per tutti i dispositivi nel bus SCSI.
- Se vengono utilizzati solo igroup con ambito backend (obsoleti a partire dalla versione 23.04), la funzione di auto-riparazione iSCSI avvierà nuove scansioni SCSI per gli ID LUN esatti nel bus SCSI.
- Se viene utilizzato un mix di igroup per nodo e igroup con ambito backend, la funzione di auto-riparazione iSCSI avvierà nuove scansioni SCSI per gli ID LUN esatti nel bus SCSI.

Installare gli strumenti iSCSI

Installa gli strumenti iSCSI utilizzando i comandi per il tuo sistema operativo.

Prima di iniziare

- Ogni nodo nel 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 il `solidfire-san` driver ed Element OS 12.5 o precedente, assicurarsi che l'algoritmo di autenticazione CHAP sia impostato su MD5 in `/etc/iscsi/iscsid.conf`. Gli algoritmi CHAP sicuri conformi a FIPS 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 worker che eseguono RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) con iSCSI PV, specificare `discard` mountOption nella StorageClass per eseguire la space reclamation inline. Consulta "Documentazione Red Hat".
- Assicurati di aver effettuato l'aggiornamento alla versione più recente del `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. Imposta la scansione su manuale:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Abilita il multipathing:

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

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

5. Assicurarsi che `iscsid` e `multipathd` siano in funzione:

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

6. Abilita e avvia `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 di open-iscsi sia 2.0.874-5ubuntu2.10 o successiva (per bionic) o 2.0.874-7.1ubuntu6.1 o successiva (per focal):

```
dpkg -l open-iscsi
```

3. Imposta la scansione su manuale:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Abilita il multipathing:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

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

5. Assicurarsi che `open-iscsi` e `multipath-tools` siano abilitati e funzionanti:

```
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 avviare `open-iscsi` affinché il demone iSCSI si avvii. In alternativa, è possibile modificare il servizio `iscsi` per avviare `iscsid` automaticamente.

Configura o disabilita l'auto-riparazione iSCSI

È possibile configurare le seguenti impostazioni di auto-riparazione Trident iSCSI per correggere le sessioni obsolete:

- **Intervallo di auto-riparazione iSCSI**: determina la frequenza con cui viene invocata l'auto-riparazione iSCSI (impostazione predefinita: 5 minuti). È possibile configurarlo per essere eseguito più frequentemente impostando un numero inferiore o meno frequentemente impostando un numero superiore.

Impostare l'intervallo di auto-riparazione iSCSI su 0 interrompe completamente l'auto-riparazione iSCSI. Non consigliamo di disabilitare l'auto-riparazione iSCSI; dovrebbe essere disabilitata solo in determinati scenari quando l'auto-riparazione iSCSI non funziona come previsto o per scopi di debug.

- **iSCSI Self-Healing Wait Time**: determina la durata che la funzionalità di auto-riparazione iSCSI attende prima di disconnettersi da una sessione non integra e tentare di accedere nuovamente (impostazione predefinita: 7 minuti). Puoi configurarlo su un numero maggiore affinché le sessioni identificate come non integre debbano attendere più a lungo prima di essere disconnesse e venga quindi effettuato un tentativo di nuovo accesso, oppure su un numero minore per disconnettersi e accedere prima.

Helm

Per configurare o modificare le impostazioni di auto-riparazione iSCSI, passare i parametri `iscsiSelfHealingInterval` e `iscsiSelfHealingWaitTime` durante l'installazione o l'aggiornamento di helm.

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

```
helm install trident trident-operator-100.2602.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

tridentctl

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

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

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

Volumi NVMe/TCP

Installare gli strumenti NVMe utilizzando i comandi per il proprio sistema operativo.

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

Verifica installazione

Dopo l'installazione, verifica che ogni nodo nel cluster Kubernetes abbia un NQN univoco utilizzando il comando:

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

Trident modifica il `ctrl_device_tmo` valore per garantire che NVMe non rinunci al percorso se va giù. Non modificare questa impostazione.

SCSI over FC volumi

Ora è possibile utilizzare il protocollo Fibre Channel (FC) con Trident per fornire e gestire risorse di storage su sistemi ONTAP.

Prerequisiti

Configurare le impostazioni di rete e dei nodi richieste per FC.

Impostazioni di rete

1. Ottieni il WWPN delle interfacce di destinazione. Consulta "network interface show" per ulteriori informazioni.
2. Ottieni il WWPN per le interfacce su initiator (Host).

   Fare riferimento alle utilità del sistema operativo host corrispondenti.

3. Configurare la suddivisione in zone sullo switch FC utilizzando i WWPN di Host e target.

   Fare riferimento alla documentazione del rispettivo switch vendor per informazioni.

   Per maggiori dettagli, fare riferimento alla seguente documentazione ONTAP:

   - "Panoramica sulla zonizzazione Fibre Channel e FCoE"

- "Modalità per configurare gli host SAN FC e FC-NVMe"

Installa gli strumenti FC

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

- Quando si utilizzano nodi worker che eseguono RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) con PV FC, specificare `discard` mountOption nel StorageClass per eseguire la space reclamation in linea. Consulta "Documentazione Red Hat".

RHEL 8+

1. Installare i seguenti pacchetti di sistema:

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

2. Abilita il multipathing:

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

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

3. Assicurati 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. Abilita il multipathing:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

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

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

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

Prepararsi al provisioning dei volumi SMB

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

> È necessario configurare entrambi i protocolli NFS e SMB/CIFS sull'SVM per creare un volume SMB `ontap-nas-economy` per i cluster ONTAP on-premises. La mancata configurazione di uno di questi protocolli causerà il fallimento della creazione del volume SMB.

> `autoExportPolicy` non è supportato per i volumi SMB.

Prima di iniziare

Per poter eseguire il provisioning dei volumi SMB, è necessario disporre di quanto segue.

- Un cluster Kubernetes con un nodo controller Linux e almeno un nodo worker Windows che esegue Windows Server 2022. Trident supporta volumi SMB montati solo su pod in esecuzione su nodi Windows.
- Almeno un secret di Trident contenente le credenziali di Active Directory. Per generare il secret `smbcreds`:

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

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

Passaggi

1. Per ONTAP on-premises, puoi facoltativamente creare una condivisione SMB oppure Trident può crearne una per te.

   > Le condivisioni SMB sono necessarie per Amazon FSx per ONTAP.

   È possibile creare le condivisioni SMB admin in due modi: utilizzando lo snap-in "Microsoft Management Console" Shared Folders o utilizzando la ONTAP CLI. Per creare le condivisioni SMB utilizzando la ONTAP CLI:

   a. Se necessario, crea la struttura del percorso della directory per la condivisione.

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

   b. Crea 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
```

Consultare "Creare una condivisione SMB" per tutti i dettagli.

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

Parametro	Descrizione	Esempio
`smbShare`	È possibile specificare uno dei seguenti valori: il nome di una condivisione SMB creata tramite Microsoft Management Console o ONTAP CLI; un nome per consentire a Trident di creare la condivisione SMB; oppure è possibile lasciare il parametro vuoto per impedire l'accesso condiviso ai volumi. Questo parametro è facoltativo per ONTAP on-premises. Questo parametro è obbligatorio per Amazon FSx for ONTAP backends e non può essere vuoto.	`smb-share`
`nasType`	**Deve essere impostato su `smb`.** Se nullo, il valore predefinito è `nfs`.	`smb`
`securityStyle`	Stile di sicurezza per i nuovi volumi. **Deve essere impostato su `ntfs` o `mixed` per i volumi SMB.**	`ntfs` o `mixed` per i volumi SMB
`unixPermissions`	Modalità per nuovi volumi. **Deve essere lasciato vuoto per i volumi SMB.**	""

Configura e gestisci i backend

Configura i backend

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

Trident offre automaticamente pool di storage da backend che soddisfano i requisiti definiti da una storage class. Scopri come configurare il backend per il tuo storage system.

- "Configura un backend Azure NetApp Files"
- "Configura un backend Google Cloud NetApp Volumes"
- "Configura un backend NetApp HCI o SolidFire"
- "Configurare un backend con driver NAS ONTAP o Cloud Volumes ONTAP"
- "Configurare un backend con driver SAN ONTAP o Cloud Volumes ONTAP"
- "Usa Trident con Amazon FSx for NetApp ONTAP"

Azure NetApp Files

Configura 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)

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 SMB	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 solo su pod in esecuzione su 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 distribuito 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"
```

Helm

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

```
helm install trident trident-operator-100.2602.0.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 distribuito utilizzando AKS
- Identità del carico di lavoro e oidc-issuer configurati sul 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
```

Helm

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)

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. Consulta "Azure: Configurare Azure NetApp Files e creare 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. Consulta "Microsoft: Crea un pool di capacità per Azure NetApp Files".
- Una sottorete delegata ad Azure NetApp Files. Consulta "Microsoft: Delegare una subnet ad Azure NetApp Files".
- `subscriptionID` da un abbonamento Azure con Azure NetApp Files abilitato.
- `tenantID`, `clientID` e `clientSecret` da un "Registrazione dell'app" in Azure Active Directory con permessi sufficienti per il servizio Azure NetApp Files. La registrazione dell'app deve utilizzare uno dei seguenti metodi:
  - Il ruolo di Owner o Contributor "predefinito da Azure".
  - Un "ruolo Contributor personalizzato" al livello di sottoscrizione (`assignableScopes`) con i seguenti permessi che sono limitati solo a ciò che Trident richiede. Dopo aver creato il ruolo personalizzato, "assegna il ruolo utilizzando il portale Azure".

Ruolo di collaboratore personalizzato

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

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

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

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

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

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

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

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

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

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

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/delete",
```

```
            "Microsoft.Features/features/read",
            "Microsoft.Features/operations/read",
            "Microsoft.Features/providers/features/read",

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

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

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

- L'`location`Azure che contiene almeno un "sottorete delegata". A partire da Trident 22.01, il parametro `location` è un campo obbligatorio al livello superiore del file di configurazione del backend. I valori di posizione specificati nei pool virtuali vengono ignorati.
- Per utilizzare `Cloud Identity`, ottenere il `client ID` da un "identità gestita assegnata all'utente" e specificare quell'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 ad Azure NetApp Files. Consulta "Microsoft: Crea e gestisci le connessioni Active Directory per Azure NetApp Files".
- Un cluster Kubernetes con un nodo controller Linux e almeno un nodo worker Windows che esegue Windows Server 2022. Trident supporta volumi SMB montati solo su pod in esecuzione su nodi Windows.
- Almeno un segreto Trident contenente le credenziali di Active Directory, così che Azure NetApp Files possa autenticarsi ad Active Directory. Per generare il secret `smbcreds`:

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

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

Opzioni di configurazione del backend di Azure NetApp Files ed esempi

Scopri le opzioni di configurazione del backend NFS e SMB per Azure NetApp Files e rivedi esempi di configurazione.

Opzioni di configurazione del backend

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 di Azure NetApp Files forniscono queste opzioni di configurazione.

Parametro	Descrizione	Predefinito
`version`	Versione della configurazione backend.	Sempre 1
`storageDriverName`	Nome del driver di archiviazione	"azure-netapp-files"
`backendName`	Nome personalizzato per il backend di storage	Driver name + "_" + caratteri casuali
`subscriptionID`	L'ID della sottoscrizione dalla tua sottoscrizione Azure. Facoltativo quando le managed identities sono abilitate su un cluster AKS.	
`tenantID`	L'ID tenant da una registrazione app. Facoltativo quando le identità gestite o l'identità cloud vengono utilizzate su un cluster AKS.	
`clientID`	L'ID client da una registrazione app. Facoltativo quando vengono utilizzate identità gestite o identità cloud su un cluster AKS.	
`clientSecret`	Il client secret da una App Registration è facoltativo quando vengono utilizzate managed identities o cloud identity 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 Facoltativo quando le identità gestite sono abilitate su un cluster AKS.	
`resourceGroups`	Elenco dei gruppi di risorse per filtrare le risorse scoperte	"[]" (nessun filtro)
`netappAccounts`	Elenco degli account NetApp per filtrare le risorse scoperte	"[]" (nessun filtro)
`capacityPools`	Elenco dei pool di capacità per filtrare le risorse scoperte	"[]" (nessun filtro, casuale)
`virtualNetwork`	Nome di una rete virtuale con una subnet delegata	""

Parametro	Descrizione	Predefinito
`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.	""
`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`	Non eseguire il provisioning se la dimensione del volume richiesto è superiore a questo valore	"" (non applicato per impostazione predefinita)
`debugTraceFlags`	Flag di debug da usare per la risoluzione dei problemi. Esempio, `\{"api": false, "method": true, "discovery": true}`. Non usare questo a meno che non si stia eseguendo una risoluzione dei problemi e sia necessario un dump dettagliato del registro.	null
`nasType`	Configura la creazione di volumi NFS o SMB. Le opzioni sono `nfs`, `smb` o null. Impostando su null, vengono creati di default volumi NFS.	`nfs`
`supportedTopologies`	Rappresenta un elenco di regioni e zone supportate da questo backend. Per ulteriori informazioni, fai riferimento a "Usa la topologia CSI".	
`qosType`	Rappresenta il tipo di QoS: Auto o Manual.	Auto

Parametro	Descrizione	Predefinito
`maxThroughput`	Imposta il throughput massimo consentito in MiB/sec. Supportato solo per i pool di capacità QoS manuali.	`4 MiB/sec`

Per ulteriori informazioni sulle Network Features, consultare "Configura le funzionalità di rete per un volume Azure NetApp Files".

Considera gli ambienti cloud di Azure (26.02)

A partire dalla versione 26.02, 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)

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.

Autorizzazioni e risorse necessarie

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 vNet 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>

Tipo	Formato
Sottorete	<resource group>/<virtual network>/<subnet>

Provisioning dei volumi

È possibile controllare il provisioning predefinito dei volumi specificando le seguenti opzioni in una sezione speciale del file di configurazione. Consultare Esempi di configurazione per i dettagli.

Parametro	Descrizione	Predefinito
`exportRule`	Regole di esportazione per i nuovi volumi. `exportRule` deve essere un elenco separato da virgole di qualsiasi combinazione di indirizzi IPv4 o sottoreti IPv4 in notazione CIDR. Ignorato per i volumi SMB.	"0.0.0.0/0"
`snapshotDir`	Accesso alla `.snapshot` directory	`true`, `false` (Impostato esplicitamente).
`size`	La dimensione predefinita dei nuovi volumi	"100G"
`unixPermissions`	I permessi unix dei nuovi volumi (4 cifre ottali). Ignorato per i volumi SMB.	"" (funzione in anteprima, richiede whitelisting nell'abbonamento)

Esempi di configurazione

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 ideale quando si sta iniziando a usare Azure NetApp Files e a fare delle prove, ma in pratica si vorrà fornire un ambito aggiuntivo per i volumi che si forniscono.

```
---
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 del backend omette `subscriptionID`, `tenantID`, `clientID` e `clientSecret`, che sono opzionali quando si usano le 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 del 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 del capacity pool

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 di backend colloca i volumi nella posizione di Azure `eastus` 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 del backend riduce ulteriormente la portata del posizionamento dei volumi a una singola subnet e modifica anche alcune impostazioni predefinite del provisioning dei volumi.

```
---
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 del pool virtuale

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, fai riferimento a "Usa 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

Le seguenti `StorageClass` definizioni si riferiscono ai pool di archiviazione sopra.

Esempi di definizioni che utilizzano `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
```

Esempi di definizioni 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 sul namespace 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 ogni namespace

```
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.

Crea il backend

Dopo aver creato il file di configurazione del backend, 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 Red Hat OpenShift
- Google Cloud NetApp Volumes pool di storage

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

Preparati 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.

Per utilizzare cloud identity con Google Cloud NetApp Volumes, è necessario disporre di:

- 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"
```

Helm

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
```

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 SMB	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 for 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

È possibile configurare Trident per eseguire il provisioning di volumi di storage a blocchi utilizzando il protocollo iSCSI da Google Cloud NetApp Volumes. I volumi SAN vengono creati dai pool di storage Flex Unified utilizzando il `google-cloud-netapp-volumes-san` driver di storage.

Questo driver è dedicato ai carichi di lavoro a blocchi e non supporta i protocolli NAS.

Il `google-cloud-netapp-volumes-san` backend è necessario per il provisioning di volumi a blocchi iSCSI. Il `google-cloud-netapp-volumes` backend supporta solo protocolli NAS e non può essere utilizzato per carichi di lavoro SAN.

Panoramica

Trident supporta i carichi di lavoro SAN (iSCSI) di Google Cloud NetApp Volumes utilizzando il driver `google-cloud-netapp-volumes-san`.

I volumi SAN vengono forniti dai pool di storage Flex Unified e presentati ai nodi Kubernetes come dispositivi a blocchi iSCSI.

Ciò si applica ai seguenti ambienti:

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

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**.

Configurare 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`.

Il tipo di file system è definito nella 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
```

Tipi di filesystem supportati:

- `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.

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
- Ridimensiona
- Importare

Comportamento di overprovisioning extra GiB

I volumi a blocchi di Google Cloud NetApp Volumes includono un overhead interno di metadati. Questo overhead riduce le dimensioni del dispositivo visibile al kernel rispetto alla capacità fornita.

I test dimostrano:

- Circa 300 KiB di overhead sulla creazione iniziale
- Fino a circa 107 MiB di overhead dopo un ridimensionamento

Poiché Google Cloud NetApp Volumes accetta solo allocazioni di interi GiB, Trident garantisce che la dimensione utilizzabile del dispositivo soddisfi o superi sempre la richiesta PVC:

- Arrotondamento della dimensione richiesta al GiB intero successivo
- Aggiunta di un buffer aggiuntivo di 1 GiB

Esempio:

- Richiesta PVC: 100 GiB
- Dimensioni fornite in Google Cloud NetApp Volumes: 101 GiB
- Spazio 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.

Prepararsi a configurare un backend Google Cloud NetApp Volumes

Prima di poter configurare il backend Google Cloud NetApp Volumes, è necessario assicurarsi 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".

Assicurarsi di disporre di quanto segue prima di configurare il backend Google Cloud NetApp Volumes:

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

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

Opzioni di configurazione del backend di Google Cloud NetApp Volumes ed esempi

Scoprite le opzioni di configurazione del backend per Google Cloud NetApp Volumes e consultate gli esempi di configurazione.

Opzioni di configurazione del backend

Ogni backend fornisce volumi in una singola regione Google Cloud. Per creare volumi in altre regioni, puoi definire backend aggiuntivi.

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di archiviazione	Il valore di `storageDriverName` deve essere specificato come "google-cloud-netapp-volumes".
`backendName`	(Facoltativo) Nome personalizzato dello storage backend	Nome driver + "_" + parte della chiave API
`storagePools`	Parametro opzionale usato per specificare i pool di storage per la creazione dei volumi.	
`projectNumber`	Numero di progetto dell'account Google Cloud. Il valore si trova nella home page del portale Google Cloud.	
`location`	La posizione di Google Cloud in cui Trident crea i volumi GCNV. Quando si creano cluster Kubernetes cross-region, i volumi creati in un `location` possono essere utilizzati nei carichi di lavoro pianificati sui nodi di più regioni di Google Cloud. Il traffico cross-region 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 del file della chiave privata di un account del servizio Google Cloud (copiato testualmente nel file di configurazione del backend). Il `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 fallisce se la dimensione del volume richiesta è superiore a questo valore.	"" (non applicato per impostazione predefinita)
`serviceLevel`	Il livello di servizio di un pool di storage e dei suoi volumi. I valori sono `flex`, `standard`, `premium`, o `extreme`.	
`labels`	Set di etichette arbitrarie in formato JSON da applicare ai volumi	""
`network`	Rete Google Cloud utilizzata per i volumi GCNV.	
`debugTraceFlags`	Flag di debug da usare per la risoluzione dei problemi. Esempio, `{"api":false, "method":true}`. Non usare questo a meno che non si stia eseguendo una risoluzione dei problemi e sia necessario un dump dettagliato del registro.	null

Parametro	Descrizione	Predefinito
`nasType`	Configura la creazione di volumi NFS o SMB. Le opzioni sono `nfs`, `smb` o null. Impostando su null, vengono creati di default volumi NFS.	`nfs`
`supportedTopologies`	Rappresenta un elenco di regioni e zone supportate da questo backend. Per ulteriori informazioni, fai riferimento a "Usa la topologia CSI". Ad esempio: `supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a`	

Opzioni di provisioning del volume

È possibile controllare il provisioning predefinito dei volumi nella sezione `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 alla `.snapshot` directory	`true`, `false` (Il comportamento predefinito potrebbe variare. Impostare esplicitamente) "false" per NFSv3
`snapshotReserve`	Percentuale di volume riservata alle snapshot	"" (accetta il valore predefinito di 0)
`unixPermissions`	I permessi unix dei nuovi volumi (4 cifre ottali).	""

Esempi di configurazione

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 scopre tutti i pool di storage delegati a Google Cloud NetApp Volumes nella posizione configurata e posiziona i nuovi volumi su uno di questi pool 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 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 i 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 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 del pool virtuale

Questa configurazione di backend definisce più pool virtuali in un unico file. I pool virtuali sono definiti nella `storage` sezione. Sono utili quando si dispone di più pool di storage che supportano diversi livelli di servizio e si desidera creare classi di storage in Kubernetes che li rappresentano. Le etichette dei pool virtuali sono utilizzate per differenziare i pool. Ad esempio, nell'esempio seguente `performance` label e `serviceLevel` type sono utilizzati per differenziare i pool virtuali.

È anche possibile impostare alcuni valori predefiniti da applicare a tutti i virtual pool e sovrascrivere i valori predefiniti per i singoli virtual pool. Nell'esempio seguente, `snapshotReserve` e `exportRule` servono come valori predefiniti per tutti i virtual pool.

Per ulteriori informazioni, fai 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

```
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 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, fai riferimento a "Usa 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
```

E ora?

Dopo aver creato il file di configurazione del backend, eseguire il seguente comando:

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

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

```
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 fallisce, c'è qualcosa di sbagliato nella configurazione del backend. Puoi descrivere il backend usando il `kubectl get tridentbackendconfig <backend-name>` comando o 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 sopra.

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

Esempi di definizioni che utilizzano il `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: 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 dettagli sulle storage class, consultare "Creare una storage class".

Esempi di definizioni 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. Per il segreto dello stage del nodo è possibile utilizzare qualsiasi utente/password Active Directory con qualsiasi o nessuna autorizzazione.

Configurazione di base sul namespace 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 ogni namespace

```
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 i pool che supportano i volumi SMB. `nasType: nfs` o `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 vincolato, 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 26.02.

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 26.02

Trident 26.02 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 26.02

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	Richiesto	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
`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 un backend NetApp HCI o SolidFire

Scoprite come creare e utilizzare un backend Element con la vostra installazione di Trident.

Dettagli del driver Element

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 storage supporta le modalità di volume *file* e *block*. Per la `Filesystem` volumeMode, Trident crea un volume e crea un filesystem. Il tipo di filesystem è 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

Avrai bisogno dei seguenti elementi prima di creare un backend Element.

- Un sistema storage supportato che esegue il software Element.
- Credenziali per un utente admin o tenant di un cluster NetApp HCI/SolidFire che può gestire i volumi.
- Tutti i nodi worker di Kubernetes devono avere installati gli strumenti iSCSI appropriati. Consulta "informazioni sulla preparazione del nodo worker".

Opzioni di configurazione del backend

Consulta la tabella seguente per le opzioni di configurazione del backend:

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di archiviazione	Sempre "solidfire-san"
`backendName`	Nome personalizzato o lo storage backend	"solidfire_" + indirizzo IP storage (iSCSI)
`Endpoint`	MVIP per il SolidFire cluster con credenziali tenant	
`SVIP`	Storage (iSCSI) indirizzo IP e porta	
`labels`	Set di etichette arbitrarie in formato JSON da applicare ai volumi.	""
`TenantName`	Nome del tenant da utilizzare (creato se non trovato)	
`InitiatorIFace`	Limitare il traffico iSCSI a una specifica interfaccia verso gli host	"default"
`UseCHAP`	Usa CHAP per autenticare iSCSI. Trident usa CHAP.	true
`AccessGroups`	Elenco degli ID dei gruppi di accesso da utilizzare	Trova l'ID di un gruppo di accesso chiamato "trident"
`Types`	Specifiche QoS	
`limitVolumeSize`	Il provisioning non riesce se la dimensione del volume richiesta è superiore a questo valore	"" (non applicato per impostazione predefinita)
`debugTraceFlags`	Flag di debug da usare per la risoluzione dei problemi. Esempio, {"api":false, "method":true}	null

ATTENZIONE Non utilizzare `debugTraceFlags` a meno che tu non stia effettuando una ricerca guasti e richieda un dump dettagliato dei registri.

Esempio 1: configurazione del backend per `solidfire-san` driver con tre tipi di volume

Questo esempio mostra un file di backend che utilizza l'autenticazione CHAP e modella tre tipi di volume con garanzie QoS specifiche. È molto probabile che si definiscano classi di storage per consumare ciascuna di queste utilizzando il parametro `IOPS` storage class.

```
---
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 storage class per `solidfire-san` driver con pool virtuali

Questo esempio mostra il file di definizione del backend configurato con pool virtuali insieme a StorageClasses che fanno riferimento ad essi.

Trident copia le etichette presenti su un pool di storage sul LUN di storage backend al momento del provisioning. Per comodità, gli amministratori dello storage possono definire etichette per pool virtuale e raggruppare i volumi in base all'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` a Silver. I pool virtuali sono definiti nella sezione `storage`. In questo esempio, alcuni dei pool di storage impostano il proprio tipo e alcuni pool sovrascrivono i valori predefiniti impostati sopra.

```
---
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 di cui sopra. Utilizzando il campo `parameters.selector`, ogni StorageClass indica quali pool virtuali possono essere utilizzati per ospitare un

volume. Il volume avrà gli aspetti definiti nel pool virtuale scelto.

Il primo StorageClass (`solidfire-gold-four`) si riferisce al primo pool virtuale. Questo è l'unico pool che offre prestazioni gold con un `Volume Type QoS` di Gold. L'ultimo StorageClass (`solidfire-silver`) richiama qualsiasi pool di storage che offre prestazioni silver. Trident deciderà quale pool virtuale selezionare e garantirà che il requisito di storage sia soddisfatto.

```
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 al volume"

Driver SAN ONTAP

Panoramica del driver ONTAP SAN

Scopri come configurare un backend ONTAP con i driver ONTAP e Cloud Volumes ONTAP SAN.

Dettagli del driver ONTAP SAN

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 nella modalità volume file system.	`xfs`, `ext3`, `ext4`
`ontap-san`	NVMe/TCP Consulta Considerazioni aggiuntive per NVMe/TCP.	Blocco	RWO, ROX, RWX, RWOP	Nessun file system; dispositivo a blocchi raw

Driver	Protocollo	volumeMode	Modalità di accesso supportate	File system supportati
`ontap-san`	NVMe/TCP Consulta Considerazioni aggiuntive per NVMe/TCP.	Filesystem	RWO, RWOP ROX e RWX non sono disponibili nella modalità volume file system.	`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 nella modalità volume file system.	`xfs`, `ext3`, `ext4`

ATTENZIONE

- Utilizzare `ontap-san-economy` solo se si prevede che il conteggio dell'utilizzo del volume persistente sia superiore a "limiti di volume ONTAP supportati".
- Utilizzare `ontap-nas-economy` solo se si prevede che il conteggio dell'utilizzo del volume persistente sia superiore a "limiti di volume ONTAP supportati" e il `ontap-san-economy` driver non può essere utilizzato.
- Non utilizzare `ontap-nas-economy` se si prevede la necessità di protezione dei dati, disaster recovery o mobilità.
- NetApp non consiglia di utilizzare l'autogrow di FlexVol in tutti i driver ONTAP, ad eccezione di ontap-san. Come soluzione alternativa, Trident supporta l'utilizzo della riserva di snapshot e ridimensiona di conseguenza i volumi FlexVol.

Permessi utente

Trident prevede di essere eseguito come amministratore ONTAP o SVM, in genere utilizzando l' `admin`utente del cluster o un `vsadmin`utente SVM, oppure un utente con un nome diverso che ha lo stesso ruolo. Per le distribuzioni Amazon FSx for NetApp ONTAP, Trident prevede di essere eseguito come amministratore ONTAP o SVM, utilizzando l'utente del cluster `fsxadmin` o un `vsadmin`utente SVM, oppure un utente con un nome diverso che ha lo stesso ruolo. L' `fsxadmin`utente è un sostituto limitato per l'utente amministratore del cluster.

NOTA

Se si utilizza il `limitAggregateUsage` parametro, sono richieste le autorizzazioni di amministratore del cluster. Quando si utilizza Amazon FSx for NetApp ONTAP con Trident, il `limitAggregateUsage` parametro non funzionerà con gli account utente `vsadmin` e `fsxadmin`. L'operazione di configurazione non andrà a buon fine se si specifica questo parametro.

Sebbene sia possibile creare un ruolo più restrittivo all'interno di ONTAP che un driver Trident può utilizzare, lo sconsigliamo. La maggior parte delle nuove versioni di Trident richiamerà API aggiuntive di cui bisognerebbe tenere conto, rendendo gli aggiornamenti difficili e soggetti a errori.

Considerazioni aggiuntive per NVMe/TCP

Trident supporta il protocollo non-volatile memory express (NVMe) utilizzando il `ontap-san` driver, incluso:

- 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
- Multipathing nativo NVMe
- Arresto regolare o non regolare dei nodi K8s (24.06)

Trident non supporta:

- DH-HMAC-CHAP che è supportato nativamente da NVMe
- Multipathing del device mapper (DM)
- Crittografia LUKS

NOTA NVMe è supportato solo con le API REST di ONTAP e non è supportato con ONTAPI (ZAPI).

Prepararsi a configurare il backend con i driver ONTAP SAN

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

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 storage. Nei sistemi ASA r2, vengono utilizzate zone di disponibilità dello storage al posto degli aggregati. Fare riferimento all'"questo"articolo della Knowledge Base su come assegnare gli aggregati alle SVM nei sistemi ASA r2.

Ricorda che puoi anche eseguire più di un driver e creare classi di archiviazione che puntano all'uno o all'altro. Ad esempio, puoi configurare una `san-dev` classe che utilizza il `ontap-san` driver e una `san-default` classe che utilizza il `ontap-san-economy` driver.

Tutti i nodi worker di Kubernetes devono avere installati gli strumenti iSCSI appropriati. Consultare "Prepara il nodo worker" per i dettagli.

Autenticare il backend ONTAP

Trident offre due modalità di autenticazione per un backend ONTAP.

- Basato su credenziali: il nome utente e la password di un utente ONTAP con le autorizzazioni richieste. Si consiglia di utilizzare un ruolo di login di sicurezza predefinito, come `admin` o `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 del 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 tra metodi basati su credenziali e metodi basati su certificati. Tuttavia, è supportato solo un metodo di autenticazione alla volta. Per passare a un diverso metodo di autenticazione, è necessario rimuovere il metodo esistente dalla configurazione del backend.

ATTENZIONE	Se si tenta di fornire **sia le credenziali che i certificati**, la creazione del backend fallirà con un errore che indica che è stato fornito più di un metodo di autenticazione nel file di configurazione.

Abilitare l'autenticazione basata su credenziali

Trident richiede le credenziali di un amministratore SVM-scoped/cluster-scoped per comunicare con il backend ONTAP. Si consiglia di utilizzare ruoli standard, predefiniti come `admin` o `vsadmin`. Questo garantisce la compatibilità futura con le versioni ONTAP che potrebbero esporre API di funzionalità da utilizzare nelle future versioni di Trident. È possibile creare e utilizzare un ruolo di accesso di sicurezza personalizzato con Trident, ma non è consigliato.

Un esempio di definizione di backend sarà 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"
}
```

Tenete presente che la definizione del backend è l'unico posto in cui le credenziali vengono archiviate in testo normale. Dopo la creazione del backend, nomi utente e password vengono codificati in Base64 e archiviati 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 riservata all'amministratore, da eseguire

dall'amministratore di Kubernetes/storage.

Abilita l'autenticazione basata sul certificato

I backend nuovi ed esistenti possono utilizzare un certificato e comunicare con il backend ONTAP. Sono richiesti tre parametri nella definizione del backend.

- clientCertificate: Valore codificato in Base64 del certificato client.
- clientPrivateKey: Valore codificato in Base64 della chiave privata associata.
- trustedCACertificate: Valore codificato in Base64 del certificato della CA fidata. Se si utilizza una CA fidata, questo parametro deve essere fornito. Questo può essere ignorato se non si utilizza una CA fidata.

Un tipico flusso di lavoro prevede i seguenti passaggi.

Passaggi

1. Generare un certificato e una chiave client. Durante la generazione, impostare Common Name (CN) sull'utente ONTAP con cui autenticarsi.

```
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. Questa operazione potrebbe essere già gestita dall'amministratore dello storage. Ignorare se non viene utilizzata una 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 punto 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 inserisci `END` per completare l'installazione.

4. Confermare che il ruolo di login di sicurezza ONTAP supporta `cert` il 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 l'autenticazione utilizzando il certificato generato. Sostituisci <ONTAP Management LIF> e <vserver name> con l'IP LIF di gestione e il 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 il certificato, la chiave e il certificato CA affidabile con Base64.

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

7. Crea il backend utilizzando i valori ottenuti nel 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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Aggiorna i metodi di autenticazione o ruota 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 nome utente/password possono essere aggiornati per utilizzare certificati; i backend che utilizzano certificati possono essere aggiornati per utilizzare nome utente/password. Per fare ciò, 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 richiesti per eseguire `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 dell'utente su ONTAP. Questo è seguito da un aggiornamento del backend. Quando si ruotano i certificati, è possibile aggiungere più certificati all'utente. Il backend viene quindi aggiornato per utilizzare il nuovo certificato, dopo di che il vecchio certificato può essere eliminato dal cluster ONTAP.

L'aggiornamento di un backend non interrompe l'accesso ai volumi già creati, né influisce sulle connessioni al volume effettuate successivamente. Un aggiornamento del backend riuscito indica che Trident può comunicare con il backend ONTAP e gestire le future operazioni sui volumi.

Crea un ruolo personalizzato ONTAP per Trident

È possibile creare un ruolo di cluster ONTAP con privilegi minimi in modo da non dover utilizzare il ruolo di amministratore ONTAP per eseguire operazioni in Trident. Quando si include il nome utente in una configurazione backend di Trident, Trident utilizza il ruolo di cluster ONTAP creato per eseguire le operazioni.

Fare riferimento a "Generatore di ruoli personalizzati Trident" per ulteriori informazioni sulla creazione di ruoli personalizzati Trident.

Utilizzo di ONTAP CLI

1. Crea un nuovo ruolo utilizzando il seguente comando:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Crea 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. Assegna 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

Eseguire i seguenti passaggi in ONTAP System Manager:

1. **Crea un ruolo personalizzato**:
   a. Per creare un ruolo personalizzato a livello di cluster, selezionare **Cluster > Settings**.

      (Oppure) Per creare un ruolo personalizzato a livello SVM, selezionare **Archiviazione > Storage VM > `required SVM`> Impostazioni > Utenti e ruoli**.

   b. Selezionare l'icona della freccia (→) accanto a **Users and Roles**.
   c. Seleziona **+Add** in **Roles**.
   d. Definisci le regole per il ruolo e fai clic su **Save**.
2. **Mappa il ruolo all'utente Trident**: + Esegui i seguenti passaggi nella pagina **Utenti e ruoli**:
   a. Selezionare l'icona Aggiungi **+** sotto **Utenti**.
   b. Selezionare il nome utente richiesto e selezionare un ruolo nel menu a discesa per **Role**.
   c. Fare clic su **Save**.

Per maggiori informazioni, consultare le seguenti pagine:

- "Ruoli personalizzati per l'amministrazione di ONTAP" o "Definisci 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` e `ontap-san-economy` driver. Ciò richiede l'abilitazione dell'opzione `useCHAP` nella definizione del backend. Quando impostato su `true`, Trident configura la sicurezza dell'initiator predefinito dell'SVM su CHAP bidirezionale e imposta il nome utente e i segreti dal file di 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` parametro è un'opzione booleana che può essere configurata una sola volta. Per impostazione predefinita, è impostato su false. Dopo averlo impostato su true, non è possibile impostarlo su false.

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

Come funziona

Impostando `useCHAP` su true, l'amministratore dello storage indica a Trident di configurare CHAP sul backend dello storage. Ciò include quanto segue:

- Configurazione di CHAP sull'SVM:
  - Se il tipo di sicurezza predefinito dell'iniziatore SVM è none (impostato per impostazione predefinita) **e** non sono presenti LUN preesistenti nel volume, Trident imposterà il tipo di sicurezza predefinito su `CHAP` e procederà alla configurazione del nome utente e dei segreti dell'iniziatore e del target CHAP.
  - Se l'SVM contiene LUN, Trident non abiliterà CHAP sull'SVM. Questo garantisce che l'accesso alle LUN già presenti sull'SVM non sia limitato.
- Configurazione del nome utente e dei segreti dell'iniziatore e del target CHAP; queste opzioni devono essere specificate nella configurazione del backend (come mostrato sopra).

Dopo la creazione del backend, Trident crea un corrispondente `tridentbackend` CRD e memorizza i segreti CHAP e i nomi utente come segreti Kubernetes. Tutti i PV 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 nel `backend.json` file. Ciò richiederà l'aggiornamento dei segreti CHAP e l'utilizzo del `tridentctl update` comando 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 ONTAP CLI o ONTAP System Manager, poiché Trident non sarà in grado di rilevare queste modifiche.

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

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

Le connessioni esistenti non saranno interessate; 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. La disconnessione e la riconnessione dei vecchi PV comporterà l'utilizzo delle credenziali aggiornate.

Opzioni ed esempi di configurazione SAN ONTAP

Scopri come creare e utilizzare i driver SAN ONTAP con la tua installazione Trident. Questa sezione fornisce esempi di configurazione del backend e dettagli per il mapping 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'utilizzo di determinati parametri come indicato. "Scopri di più sulle differenze tra i sistemi ASA r2 e gli altri sistemi ONTAP". Nella configurazione del

backend Trident, non è necessario specificare che il sistema sia ASA r2. Quando si seleziona `ontap-san` come `storageDriverName`, Trident rileva automaticamente i sistemi 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` driver (con protocolli iSCSI, NVMe/TCP e FC) è supportato per i sistemi ASA r2.

Opzioni di configurazione del backend

Consulta la tabella seguente per le opzioni di configurazione del backend:

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di archiviazione	`ontap-san` o `ontap-san-economy`
`backendName`	Nome personalizzato o lo storage backend	Nome driver + "_" + dataLIF
`managementLIF`	Indirizzo IP di un cluster o di una LIF di gestione SVM. È possibile specificare un domain name pienamente qualificato (FQDN). Può essere impostato per usare indirizzi IPv6 se hai installato Trident usando il flag IPv6. Gli indirizzi IPv6 devono essere definiti tra parentesi quadre, ad esempio `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Per un passaggio senza interruzioni di MetroCluster, vedere Esempio di MetroCluster. **NOTA** Se si utilizzano le credenziali "vsadmin", `managementLIF` deve essere quella dell'SVM; se si utilizzano le credenziali "admin", `managementLIF` deve essere quella del cluster.	"10.0.0.1", "[2001:1234:abcd::fefe]"

Parametro	Descrizione	Predefinito
`dataLIF`	Indirizzo IP del protocollo LIF. Può essere impostato per usare indirizzi IPv6 se hai installato Trident usando 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 "ONTAP Selective LUN Map" per rilevare i LIF iSCSI necessari per stabilire una sessione multipath. Viene generato un avviso se `dataLIF` è definito esplicitamente. **Omettere per MetroCluster.** Vedere il Esempio di MetroCluster.	Derivato dall'SVM
`svm`	Macchina virtuale di storage da utilizzare **Ometti per MetroCluster.** Vedere il Esempio di MetroCluster.	Derivato se viene specificato un SVM `managementLIF`
`useCHAP`	Utilizzare CHAP per autenticare iSCSI per i driver SAN ONTAP [Booleano]. Impostare su `true` per consentire a Trident di configurare e utilizzare CHAP bidirezionale come autenticazione predefinita per l'SVM specificato nel backend. Consultare "Prepararsi a configurare il backend con i driver ONTAP SAN" per i dettagli. **Non supportato per FCP o NVMe/TCP.**	`false`
`chapInitiatorSecret`	Segreto dell'iniziatore CHAP. Obbligatorio se `useCHAP=true`	""
`labels`	Set di etichette arbitrarie in formato JSON da applicare ai volumi	""
`chapTargetInitiatorSecret`	Segreto dell'iniziatore di destinazione CHAP. Obbligatorio se `useCHAP=true`	""
`chapUsername`	Nome utente in entrata. Obbligatorio se `useCHAP=true`	""
`chapTargetUsername`	Nome utente di destinazione. Obbligatorio se `useCHAP=true`	""
`clientCertificate`	Valore codificato in Base64 del certificato client. 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. Facoltativo. Utilizzato per l'autenticazione basata su certificato.	""
`username`	Nome utente necessario per comunicare con il cluster ONTAP. Utilizzata per l'autenticazione basata sulle credenziali. Per l'autenticazione Active Directory, vedere "Autenticare Trident a un backend SVM utilizzando le credenziali di Active Directory".	""

Parametro	Descrizione	Predefinito
`password`	Password necessaria per comunicare con il cluster ONTAP. Utilizzata per l'autenticazione basata sulle credenziali. Per l'autenticazione Active Directory, vedere "Autenticare Trident a un backend SVM utilizzando le credenziali di Active Directory".	""
`svm`	Macchina virtuale di storage da utilizzare	Derivato se viene specificato un SVM `managementLIF`
`storagePrefix`	Prefisso utilizzato durante il provisioning di nuovi volumi nell'SVM. Non può essere modificato in seguito. Per aggiornare questo parametro, sarà necessario creare un nuovo backend.	`trident`
`aggregate`	Aggregato per il provisioning (facoltativo; se impostato, deve essere assegnato alla SVM). Per il `ontap-nas-flexgroup` driver, questa opzione viene ignorata. Se non assegnato, qualsiasi degli aggregati disponibili può essere utilizzato per il provisioning di un FlexGroup volume. **NOTA** Quando l'aggregato viene aggiornato in SVM, viene aggiornato automaticamente in Trident interrogando 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 passerà allo stato di errore in Trident durante l'interrogazione dell'aggregato SVM. Devi modificare l'aggregato con uno presente sulla SVM o rimuoverlo completamente per riportare online il backend. **Non specificare per i sistemi ASA r2**.	""
`limitAggregateUsage`	Il provisioning fallisce se l'utilizzo supera questa percentuale. Se si utilizza un Amazon FSx for NetApp ONTAP backend, non specificare `limitAggregateUsage`. I `fsxadmin` e `vsadmin` forniti non contengono le autorizzazioni necessarie per recuperare l'utilizzo aggregato e limitarlo tramite Trident. **Non specificare per i sistemi ASA r2**.	"" (non applicato per impostazione predefinita)
`limitVolumeSize`	Il provisioning fallisce se la dimensione del volume richiesto supera questo valore. Limita anche la dimensione massima dei volumi che gestisce per LUN.	"" (non applicato per impostazione predefinita)

Parametro	Descrizione	Predefinito
`lunsPerFlexvol`	Numero massimo di LUN per FlexVol, deve essere nell'intervallo [50, 200]	`100`
`debugTraceFlags`	Flag di debug da usare per la risoluzione dei problemi. Esempio, {"api":false, "method":true} non usare a meno che non si stia eseguendo una risoluzione dei problemi e sia necessario un dump dettagliato del registro.	`null`
`useREST`	Parametro booleano per utilizzare le API REST di ONTAP. `` `useREST` Quando impostato su `true`, Trident utilizza le ONTAP REST APIs per comunicare con il backend; quando impostato su `false`, Trident utilizza le chiamate ONTAPI (ZAPI) per comunicare con il backend. Questa funzione richiede ONTAP 9.11.1 e versioni successive. Inoltre, il ruolo di login ONTAP utilizzato deve avere accesso all'applicazione `ontapi`. Questo è soddisfatto dai ruoli predefiniti `vsadmin` e `cluster-admin`. A partire dalla release Trident 24.06 e ONTAP 9.15.1 o versioni successive, `useREST` è impostato su `true` per impostazione predefinita; modificare `useREST` su `false` per utilizzare le chiamate ONTAPI (ZAPI). `` `useREST` è pienamente qualificato per NVMe/TCP. **NOTA** NVMe è supportato solo con le API REST di ONTAP e non è supportato con ONTAPI (ZAPI). **Se specificato, impostare sempre su `true` per i sistemi ASA r2**.	`true` per ONTAP 9.15.1 o versioni successive, altrimenti `false`.
`sanType`	Utilizzare per selezionare `iscsi` per iSCSI, `nvme` per NVMe/TCP o `fcp` per 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 saranno applicati ogni volta che un volume viene formattato. Questo consente di formattare il volume secondo le tue preferenze. Assicurati di specificare i formatOptions simili a quelli delle 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, supportato per sistemi ASA r2 quando si utilizzano i protocolli iSCSI e NVMe/TCP.**	
`limitVolumePoolSize`	Dimensione massima richiedibile FlexVol quando si usano LUN nel backend ontap-san-economy.	"" (non applicato per impostazione predefinita)
`denyNewVolumePools`	Restringe i backend dal `ontap-san-economy` creare nuovi volumi FlexVol per contenere le loro LUN. Solo i FlexVol preesistenti vengono utilizzati per il provisioning di nuovi PV.	

Raccomandazioni per l'utilizzo di formatOptions

Trident raccomanda le seguenti opzioni per accelerare il processo di formattazione:

- **-E nodiscard (ext3, ext4):** Non tentare di scartare i blocchi al momento di mkfs (scartare i blocchi inizialmente è utile sui dispositivi a stato solido e sullo storage sparse / con thin provisioning). Questo sostituisce l'opzione deprecata "-K" ed è applicabile ai file system ext3, ext4.
- **-K (xfs):** Non tentare di scartare i blocchi al momento di mkfs. Questa opzione è applicabile al file system xfs.

Autenticare Trident a un backend SVM utilizzando le credenziali di Active Directory

È possibile configurare Trident per autenticarsi a un backend SVM utilizzando le credenziali di Active Directory (AD). Prima che un account AD possa accedere all'SVM, è necessario configurare l'accesso del domain controller AD al cluster o all'SVM. Per l'amministrazione del cluster con un account AD, è necessario creare un domain tunnel. Consultare "Configurare l'accesso del domain controller Active Directory in ONTAP" per i dettagli.

passi

1. Configurare le impostazioni del Domain Name System (DNS) per una SVM di 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 la 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 i parametri `username` e `password` rispettivamente sul nome utente o del gruppo AD e sulla password.

Opzioni di configurazione del backend per il provisioning dei volumi

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

Parametro	Descrizione	Predefinito
`spaceAllocation`	Allocazione dello spazio per le LUN	"true" **Se specificato, impostare su `true` per sistemi ASA r2**.
`spaceReserve`	Modalità di prenotazione dello spazio; "none" (con thin provisioning) o "volume" (con thick provisioning). **Impostare su `none` per sistemi ASA r2**.	"none"
`snapshotPolicy`	Policy di Snapshot da utilizzare. **Impostare su `none` per sistemi ASA r2**.	"none"
`qosPolicy`	Gruppo di policy QoS da assegnare ai volumi creati. Scegliere uno tra qosPolicy o adaptiveQosPolicy per pool di storage/backend. L'utilizzo di gruppi di policy QoS con Trident richiede ONTAP 9.8 o versioni successive. È consigliabile utilizzare un gruppo di policy QoS non condiviso e assicurarsi che il gruppo di policy venga applicato a ciascun componente singolarmente. Un gruppo di policy QoS condiviso impone il limite massimo per il throughput di tutti i carichi di lavoro.	""
`adaptiveQosPolicy`	Gruppo di policy QoS adattivo da assegnare ai volumi creati. Scegliere uno tra qosPolicy o adaptiveQosPolicy per storage pool/backend	""
`snapshotReserve`	Percentuale di volume riservata alle snapshot. **Non specificare per i sistemi ASA r2**.	"0" se `snapshotPolicy` è "none", altrimenti ""
`splitOnClone`	Dividere un clone dal suo genitore al momento della creazione	"false"
`encryption`	Abilita 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 fornito in Trident sarà abilitato per NAE. Per ulteriori informazioni, fare riferimento a: "Come funziona Trident con NVE e NAE".	"false" **Se specificato, impostare su `true` per i sistemi ASA r2**.

Parametro	Descrizione	Predefinito
`luksEncryption`	Abilita la crittografia LUKS. Consulta "Usa Linux Unified Key Setup (LUKS)".	"" **Impostato su `false` per i sistemi ASA r2.**
`tieringPolicy`	Criterio di tiering da utilizzare "none" **Non specificare per i sistemi ASA r2.**	
`nameTemplate`	Modello per creare nomi di volume personalizzati.	""

Esempi di provisioning del volume

Ecco un esempio con i valori predefiniti:

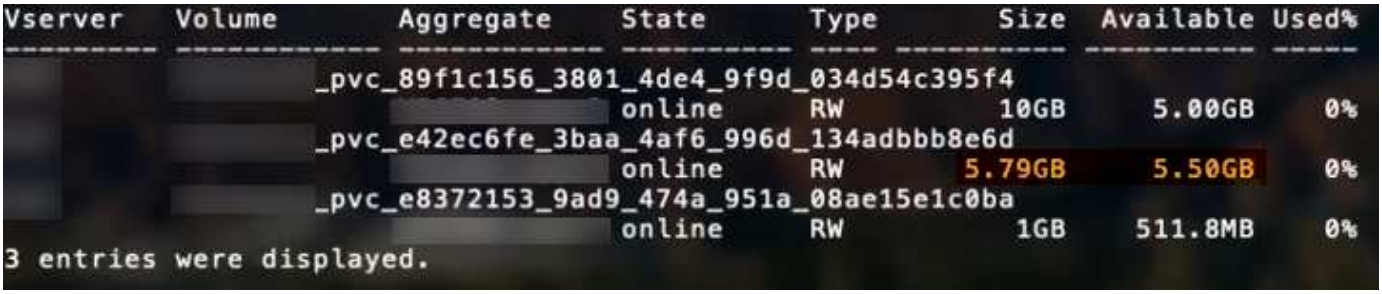
```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

NOTA Per tutti i volumi creati utilizzando il driver `ontap-san`, Trident aggiunge un ulteriore 10 percento di capacità alla FlexVol per ospitare i metadati della LUN. La LUN verrà fornita con la dimensione esatta richiesta dall'utente nel PVC. Trident aggiunge il 10 percento alla FlexVol (visualizzato come Available size in ONTAP). Gli utenti otterranno ora la quantità di capacità utilizzabile richiesta. Questa modifica impedisce anche che le 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 la dimensione 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 che Trident aggiunge a FlexVol per ospitare i metadati della LUN. Per `snapshotReserve` = 5% e PVC request = 5 GiB, la dimensione totale del volume è 5,79 GiB e la dimensione disponibile è 5,5 GiB. Il comando `volume show` dovrebbe mostrare risultati simili a questo esempio:



Vserver	Volume	Aggregate	State	Type	Size	Available	Used%
	_pvc_89f1c156_3801_4de4_9f9d_034d54c395f4						
			online	RW	10GB	5.00GB	0%
	_pvc_e42ec6fe_3baa_4af6_996d_134adbbb8e6d						
			online	RW	5.79GB	5.50GB	0%
	_pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba						
			online	RW	1GB	511.8MB	0%

3 entries were displayed.

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

Esempi di configurazione minima

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.

NOTA	Se si utilizza Amazon FSx su NetApp ONTAP con Trident, NetApp consiglia di specificare i nomi DNS per i LIF invece degli indirizzi IP.

Esempio SAN ONTAP

Questa è una configurazione di base che utilizza il `ontap-san` driver.

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

Esempio di 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 SVM".

Per un passaggio e un ritorno senza interruzioni, specificare l'SVM utilizzando `managementLIF` e omettere 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 popolati in `backend.json` e accettano rispettivamente i valori codificati in 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 di CHAP bidirezionale

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

Esempio ONTAP SAN 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 CHAP economy ONTAP SAN

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

Esempio NVMe/TCP

È necessario disporre di una SVM configurata con NVMe sul backend ONTAP. Questa è una configurazione di base del backend 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)

È necessario disporre di una SVM configurata con FC sul backend ONTAP. Questa è una configurazione di base del backend 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 del 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}}"
```

formatOptions example 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 backend di esempio, vengono impostati valori predefiniti specifici per tutti i pool di storage, come `spaceReserve` a none, `spaceAllocation` a false e `encryption` a false. I pool virtuali sono definiti nella sezione storage.

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

In questi esempi, alcuni pool di storage impostano i propri `spaceReserve`, `spaceAllocation` e `encryption` valori, mentre alcuni pool sovrascrivono i valori predefiniti.

Esempio 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"
```

Mappa i backend a 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 avrà gli aspetti definiti nel pool virtuale scelto.

- Il `protection-gold` StorageClass verrà mappato sul primo pool virtuale nel `ontap-san` backend. Questo è l'unico pool che offre protezione di livello gold.

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

- Il `protection-not-gold` StorageClass verrà mappato sul secondo e 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"
```

- Il `app-mysqldb` StorageClass verrà mappato sul terzo pool virtuale nel `ontap-san-economy` backend. Questo è l'unico pool che offre la configurazione dello storage pool per l'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"
```

- Il `protection-silver-creditpoints-20k` StorageClass verrà mappato sul 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"
```

- Il `creditpoints-5k` StorageClass verrà mappato sul terzo pool virtuale nel `ontap-san` backend e sul 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"
```

- Il `my-test-app-sc` StorageClass verrà mappato 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 deciderà quale pool virtuale selezionare e garantirà che il requisito di storage sia soddisfatto.

Driver NAS ONTAP

Panoramica del driver ONTAP NAS

Scopri come configurare un backend ONTAP con i driver ONTAP e Cloud Volumes ONTAP NAS.

Dettagli del driver ONTAP NAS

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 SMB	Filesystem	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`
`ontap-nas-economy`	NFS SMB	Filesystem	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`
`ontap-nas-flexgroup`	NFS SMB	Filesystem	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

ATTENZIONE

- Utilizzare `ontap-san-economy` solo se si prevede che il conteggio dell'utilizzo del volume persistente sia superiore a "limiti di volume ONTAP supportati".
- Utilizzare `ontap-nas-economy` solo se si prevede che il conteggio dell'utilizzo del volume persistente sia superiore a "limiti di volume ONTAP supportati" e il `ontap-san-economy` driver non può essere utilizzato.
- Non utilizzare `ontap-nas-economy` se si prevede la necessità di protezione dei dati, disaster recovery o mobilità.
- NetApp non consiglia di utilizzare l'autogrow di FlexVol in tutti i driver ONTAP, ad eccezione di ontap-san. Come soluzione alternativa, Trident supporta l'utilizzo della riserva di snapshot e ridimensiona di conseguenza i volumi FlexVol.

Permessi utente

Trident prevede di essere eseguito come amministratore ONTAP o SVM, in genere utilizzando l' `admin`utente del cluster o un `vsadmin`utente SVM, oppure un utente con un nome diverso che ha lo stesso ruolo.

Per le distribuzioni Amazon FSx for NetApp ONTAP, Trident prevede di essere eseguito come amministratore ONTAP o SVM, utilizzando l'utente del cluster `fsxadmin` o un `vsadmin`utente SVM, oppure un utente con un nome diverso che ha lo stesso ruolo. L' `fsxadmin`utente è un sostituto limitato per l'utente amministratore del cluster.

NOTA

Se si utilizza il `limitAggregateUsage` parametro, sono richieste le autorizzazioni di amministratore del cluster. Quando si utilizza Amazon FSx for NetApp ONTAP con Trident, il `limitAggregateUsage` parametro non funzionerà con gli account utente `vsadmin` e `fsxadmin`. L'operazione di configurazione non andrà a buon fine se si specifica questo parametro.

Sebbene sia possibile creare un ruolo più restrittivo all'interno di ONTAP che un driver Trident può utilizzare, lo sconsigliamo. La maggior parte delle nuove versioni di Trident richiamerà API aggiuntive di cui bisognerebbe tenere conto, rendendo gli aggiornamenti difficili e soggetti a errori.

Prepararsi a configurare un backend con i driver ONTAP NAS

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 "NetApp AFX sistema storage". 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` 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 archiviazione che puntano all'uno o all'altro. Ad esempio, si può configurare una classe Gold che utilizza il `ontap-nas` driver e una classe Bronze che

utilizza il `ontap-nas-economy` driver.

- Su tutti i nodi worker di Kubernetes devono essere installati gli strumenti NFS appropriati. Consultare "qui" per maggiori dettagli.
- Trident supporta volumi SMB montati solo su pod in esecuzione su nodi Windows. Consultare Prepararsi al provisioning dei volumi SMB per i dettagli.

Autenticare il backend ONTAP

Trident offre due modalità di autenticazione per un backend ONTAP.

- Basato su credenziali: Questa modalità richiede autorizzazioni sufficienti al backend ONTAP. Si consiglia di utilizzare un account associato a un ruolo di login di sicurezza predefinito, come `admin` o `vsadmin` per garantire la massima compatibilità con le versioni ONTAP.
- Basato su certificato: Questa modalità richiede un certificato installato sul backend per Trident per comunicare con un cluster ONTAP. In questo caso, la definizione del 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 tra metodi basati su credenziali e metodi basati su certificati. Tuttavia, è supportato solo un metodo di autenticazione alla volta. Per passare a un diverso metodo di autenticazione, è necessario rimuovere il metodo esistente dalla configurazione del backend.

ATTENZIONE	Se si tenta di fornire **sia le credenziali che i certificati**, la creazione del backend fallirà con un errore che indica che è stato fornito più di un metodo di autenticazione nel file di configurazione.

Abilitare l'autenticazione basata su credenziali

Trident richiede le credenziali di un amministratore SVM-scoped/cluster-scoped per comunicare con il backend ONTAP. Si consiglia di utilizzare ruoli standard, predefiniti come `admin` o `vsadmin`. Questo garantisce la compatibilità futura con le versioni ONTAP che potrebbero esporre API di funzionalità da utilizzare nelle future versioni di Trident. È possibile creare e utilizzare un ruolo di accesso di sicurezza personalizzato con Trident, ma non è consigliato.

Un esempio di definizione di backend sarà 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"
    }
}
```

Tenete presente che la definizione del backend è l'unico posto in cui le credenziali vengono archiviate in testo normale. Dopo la creazione del backend, nomi utente e password vengono codificati in Base64 e archiviati come segreti Kubernetes. La creazione/aggiornamento di un backend è l'unico passaggio che richiede la conoscenza delle credenziali. Pertanto, si tratta di un'operazione riservata all'amministratore, da eseguire dall'amministratore di Kubernetes/storage.

Abilita l'autenticazione basata su certificati

I backend nuovi ed esistenti possono utilizzare un certificato e comunicare con il backend ONTAP. Sono richiesti tre parametri nella definizione del backend.

- clientCertificate: Valore codificato in Base64 del certificato client.
- clientPrivateKey: Valore codificato in Base64 della chiave privata associata.
- trustedCACertificate: Valore codificato in Base64 del certificato della CA fidata. Se si utilizza una CA fidata, questo parametro deve essere fornito. Questo può essere ignorato se non si utilizza una CA fidata.

Un tipico flusso di lavoro prevede i seguenti passaggi.

Passaggi

1. Generare un certificato e una chiave client. Durante la generazione, impostare Common Name (CN)

sull'utente ONTAP con cui autenticarsi.

```
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. Questa operazione potrebbe essere già gestita dall'amministratore dello storage. Ignorare se non viene utilizzata una 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 punto 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. Confermare che il ruolo di login di sicurezza ONTAP supporta `cert` il 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 l'autenticazione utilizzando il certificato generato. Sostituisci <ONTAP Management LIF> e <vserver name> con l'IP LIF di gestione e il nome SVM. È necessario assicurarsi che il LIF abbia la sua service policy 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 il certificato, la chiave e il certificato CA affidabile con Base64.

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

7. Crea il backend utilizzando i valori ottenuti nel 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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Aggiorna i metodi di autenticazione o ruota 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 nome utente/password possono essere aggiornati per utilizzare certificati; i backend che utilizzano certificati possono essere aggiornati per utilizzare nome utente/password. Per fare ciò, 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 richiesti per eseguire `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 dell'utente su ONTAP. Questo è seguito da un aggiornamento del backend. Quando si ruotano i certificati, è possibile aggiungere più certificati all'utente. Il backend viene quindi aggiornato per utilizzare il nuovo certificato, dopo di che il vecchio certificato può essere eliminato dal cluster ONTAP.

L'aggiornamento di un backend non interrompe l'accesso ai volumi già creati, né influisce sulle connessioni al volume effettuate successivamente. Un aggiornamento del backend riuscito indica che Trident può comunicare con il backend ONTAP e gestire le future operazioni sui volumi.

Crea un ruolo personalizzato ONTAP per Trident

È possibile creare un ruolo di cluster ONTAP con privilegi minimi in modo da non dover utilizzare il ruolo di amministratore ONTAP per eseguire operazioni in Trident. Quando si include il nome utente in una configurazione backend di Trident, Trident utilizza il ruolo di cluster ONTAP creato per eseguire le operazioni.

Fare riferimento a "Generatore di ruoli personalizzati Trident" per ulteriori informazioni sulla creazione di ruoli personalizzati Trident.

Utilizzo di ONTAP CLI

1. Crea un nuovo ruolo utilizzando il seguente comando:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Crea 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. Assegna 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

Eseguire i seguenti passaggi in ONTAP System Manager:

1. **Crea un ruolo personalizzato**:
   a. Per creare un ruolo personalizzato a livello di cluster, selezionare **Cluster > Settings**.

      (Oppure) Per creare un ruolo personalizzato a livello SVM, selezionare **Archiviazione > Storage VM > `required SVM`> Impostazioni > Utenti e ruoli**.

   b. Selezionare l'icona della freccia (→) accanto a **Users and Roles**.
   c. Seleziona **+Add** in **Roles**.
   d. Definisci le regole per il ruolo e fai clic su **Save**.
2. **Mappa il ruolo all'utente Trident**: + Esegui i seguenti passaggi nella pagina **Utenti e ruoli**:
   a. Selezionare l'icona Aggiungi **+** sotto **Utenti**.
   b. Selezionare il nome utente richiesto e selezionare un ruolo nel menu a discesa per **Role**.
   c. Fare clic su **Save**.

Per maggiori informazioni, consultare le seguenti pagine:

- "Ruoli personalizzati per l'amministrazione di ONTAP" o "Definisci ruoli personalizzati"
- "Lavorare con ruoli e utenti"

Gestisci le policy di esportazione NFS

Trident utilizza le policy di esportazione NFS per controllare l'accesso ai volumi che fornisce.

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

- Trident può gestire dinamicamente la policy di esportazione stessa; in questa modalità operativa, l'amministratore dello storage specifica un elenco di blocchi CIDR che rappresentano gli indirizzi IP

ammissibili. Trident aggiunge automaticamente gli IP dei nodi che rientrano in questi intervalli alla policy di esportazione al momento della pubblicazione. In alternativa, quando non vengono specificati CIDR, tutti gli IP unicast a livello globale trovati sul nodo a cui si sta pubblicando il volume verranno aggiunti alla policy di esportazione.

- Gli amministratori dello storage possono creare una policy di esportazione e aggiungere regole manualmente. Trident utilizza la policy di esportazione predefinita a meno che nella configurazione non venga specificato un nome diverso per la policy di esportazione.

Gestisci dinamicamente le policy di esportazione

Trident offre la possibilità di gestire dinamicamente le policy di esportazione per i backend ONTAP. Questo offre all'amministratore dello storage la possibilità di specificare uno spazio di indirizzi consentito per gli IP dei nodi worker, invece di definire manualmente regole esplicite. Questo semplifica notevolmente la gestione delle policy di esportazione; le modifiche alla policy di esportazione non richiedono più l'intervento manuale sul cluster di storage. Inoltre, questo aiuta a limitare l'accesso al cluster di storage solo ai nodi worker che stanno montando i volumi e hanno IP nell'intervallo specificato, supportando una gestione automatizzata e a grana fine.

NOTA	Non utilizzare il Network Address Translation (NAT) quando si usano le policy di esportazione dinamiche. Con il NAT, lo storage controller vede l'indirizzo NAT del frontend e non l'indirizzo IP effettivo dell'host, quindi l'accesso sarà negato quando non viene trovata alcuna corrispondenza nelle regole di esportazione.

Esempio

Ci sono due opzioni di configurazione che devono essere utilizzate. Ecco un esempio di definizione 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 principale nell'SVM abbia una policy di esportazione precedentemente creata con una regola di esportazione che permetta il blocco CIDR del nodo (ad esempio la policy di esportazione predefinita). Seguire sempre la best practice raccomandata da NetApp di dedicare un SVM per Trident.

Ecco una spiegazione di come funziona questa funzione utilizzando l'esempio sopra:

- `autoExportPolicy` è impostato su `true`. Questo indica che Trident crea una policy di esportazione per ogni volume fornito con questo backend per la SVM `svm1` e gestisce l'aggiunta e l'eliminazione delle

regole utilizzando blocchi di indirizzi `autoexportCIDRs`. Fino a quando un volume non è collegato a un nodo, il volume utilizza una policy di esportazione vuota senza regole per impedire accessi indesiderati a quel volume. Quando un volume viene pubblicato su 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 anche aggiunti alla policy di esportazione utilizzata dal volume FlexVol padre

    - Ad esempio:
        - UUID backend 403b5326-8482-40db-96d0-d83fb3f4daec
        - `autoExportPolicy` impostato su `true`
        - prefisso storage `trident`
        - PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
        - Il qtree denominato trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c crea una policy di esportazione per il FlexVol denominato `trident-403b5326-8482-40db96d0-d83fb3f4daec`, una policy di esportazione per il qtree denominato `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c` e una policy di esportazione vuota denominata `trident_empty` sull'SVM. Le regole per la policy di esportazione del FlexVol saranno un superset di tutte le regole contenute nelle policy di esportazione dei qtree. La policy di esportazione vuota sarà riutilizzata da tutti i volumi che non sono collegati.

- `autoExportCIDRs` contiene un elenco di blocchi di indirizzi. Questo campo è opzionale e il valore predefinito è ["0.0.0.0/0", "::/0"]. Se non è definito, Trident aggiunge tutti gli indirizzi unicast con ambito globale trovati sui nodi worker con pubblicazioni.

In questo esempio, viene fornito lo spazio di indirizzi `192.168.0.0/24`. Ciò indica che gli IP dei nodi Kubernetes che rientrano in questo intervallo di indirizzi con pubblicazioni saranno aggiunti alla export policy che Trident crea. Quando Trident registra un nodo su cui viene eseguito, recupera gli indirizzi IP del nodo e li controlla rispetto ai blocchi di indirizzi forniti in `autoExportCIDRs`. Al momento della pubblicazione, dopo aver filtrato gli IP, Trident crea le regole di export policy per gli IP client del nodo su cui sta pubblicando.

È possibile aggiornare `autoExportPolicy` e `autoExportCIDRs` per i backend dopo averli creati. È possibile aggiungere nuovi CIDR per un backend gestito automaticamente o eliminare i CIDR esistenti. Prestare attenzione quando si eliminano i CIDR per garantire che le connessioni esistenti non vengano interrotte. È anche possibile scegliere di disabilitare `autoExportPolicy` per un backend e tornare a una policy di esportazione creata manualmente. Questo richiederà l'impostazione del parametro `exportPolicy` nella configurazione del backend.

Dopo che Trident ha creato o aggiornato un backend, puoi verificare il backend utilizzando `tridentctl` o il corrispondente `tridentbackend` CRD:

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

Quando un nodo viene rimosso, Trident controlla tutte le policy di esportazione per rimuovere le regole di accesso corrispondenti al nodo. Rimuovendo questo IP del 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 già esistenti, l'aggiornamento del backend con `tridentctl update backend` garantisce che Trident gestisca automaticamente le policy di esportazione. Questo crea due nuove policy di esportazione denominate in base all'UUID del backend e al nome del qtree quando necessario. I volumi presenti sul backend utilizzeranno le nuove policy di esportazione dopo essere stati smontati e rimontati.

NOTA	L'eliminazione di un backend con policy di esportazione gestite automaticamente eliminerà la policy di esportazione creata dinamicamente. Se il backend viene ricreato, viene trattato come un nuovo backend e comporterà la creazione di una nuova policy di esportazione.

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

Prepararsi al provisioning dei volumi SMB

Con una piccola preparazione aggiuntiva, è possibile eseguire il provisioning dei volumi SMB utilizzando `ontap-nas` driver.

ATTENZIONE	È necessario configurare entrambi i protocolli NFS e SMB/CIFS sull'SVM per creare un volume SMB `ontap-nas-economy` per i cluster ONTAP on-premises. La mancata configurazione di uno di questi protocolli causerà il fallimento della creazione del volume SMB.

NOTA `autoExportPolicy` non è supportato per i volumi SMB.

Prima di iniziare

Per poter eseguire il provisioning dei volumi SMB, è necessario disporre di quanto segue.

- Un cluster Kubernetes con un nodo controller Linux e almeno un nodo worker Windows che esegue Windows Server 2022. Trident supporta volumi SMB montati solo su pod in esecuzione su nodi Windows.
- Almeno un secret di Trident contenente le credenziali di Active Directory. Per generare il secret `smbcreds`:

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

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

Passaggi

1. Per ONTAP on-premises, puoi facoltativamente creare una condivisione SMB oppure Trident può crearne una per te.

   NOTA Le condivisioni SMB sono necessarie per Amazon FSx per ONTAP.

   È possibile creare le condivisioni SMB admin in due modi: utilizzando lo snap-in "Microsoft Management Console" Shared Folders o utilizzando la ONTAP CLI. Per creare le condivisioni SMB utilizzando la ONTAP CLI:

   a. Se necessario, crea la struttura del percorso della directory per la condivisione.

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

   b. Crea 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 Consultare "Creare una condivisione SMB" per tutti i dettagli.

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

Parametro	Descrizione	Esempio
`smbShare`	È possibile specificare uno dei seguenti valori: il nome di una condivisione SMB creata tramite Microsoft Management Console o ONTAP CLI; un nome per consentire a Trident di creare la condivisione SMB; oppure è possibile lasciare il parametro vuoto per impedire l'accesso condiviso ai volumi. Questo parametro è facoltativo per ONTAP on-premises. Questo parametro è obbligatorio per Amazon FSx for ONTAP backends e non può essere vuoto.	`smb-share`
`nasType`	**Deve essere impostato su `smb`.** Se nullo, il valore predefinito è `nfs`.	`smb`
`securityStyle`	Stile di sicurezza per i nuovi volumi. **Deve essere impostato su `ntfs` o `mixed` per i volumi SMB.**	`ntfs` o `mixed` per i volumi SMB
`unixPermissions`	Modalità per nuovi volumi. **Deve essere lasciato vuoto per i volumi SMB.**	""

Abilita SMB sicuro

A partire dalla release 25.06, NetApp Trident supporta il provisioning sicuro dei volumi SMB creati utilizzando `ontap-nas` e `ontap-nas-economy` backends. Quando è abilitato l'SMB sicuro, è possibile fornire un accesso controllato alle condivisioni SMB per gli utenti e i gruppi di utenti di Active Directory (AD) utilizzando le Access Control Lists (ACLs).

Punti da ricordare

- L'importazione `ontap-nas-economy` di volumi non è supportata.
- Sono supportati solo i cloni di sola lettura per i volumi `ontap-nas-economy`.
- Se Secure SMB è abilitato, Trident ignorerà la condivisione SMB menzionata nel backend.
- L'aggiornamento dell'annotazione PVC, dell'annotazione storage class e del campo backend non aggiorna l'ACL della condivisione SMB.
- Le ACL di condivisione SMB specificate nell'annotazione del PVC clone avranno la precedenza su quelle nel PVC di origine.
- Assicurati di fornire utenti AD validi quando abiliti SMB sicuro. Gli utenti non validi non verranno aggiunti all'ACL.
- Se si fornisce lo stesso utente AD nel backend, storage class e PVC con autorizzazioni diverse, la priorità delle autorizzazioni sarà: PVC, storage class e poi backend.
- Secure SMB è supportato per `ontap-nas` le importazioni di volumi gestiti e non è applicabile alle importazioni di volumi non gestiti.

Passaggi

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. Aggiungi l'annotazione nella storage class.

   Aggiungere l'annotazione `trident.netapp.io/smbShareAdUser` alla storage class per abilitare SMB sicuro senza errori. Il valore utente specificato per l'annotazione `trident.netapp.io/smbShareAdUser` deve essere lo stesso del nome utente specificato nel `smbcreds` secret. È possibile scegliere una delle seguenti opzioni per `smbShareAdUserPermission`: `full_control`, `change` o `read`. L'autorizzazione predefinita 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. Crea un PVC.

   Il seguente esempio 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 NAS ONTAP

Impara a creare e utilizzare i driver NAS ONTAP con la tua installazione Trident. Questa sezione fornisce esempi di configurazione del backend e dettagli per il mapping dei backend a StorageClasses. A partire dalla release 25.10, NetApp Trident supporta "Sistemi storage NetApp AFX". NetApp AFX storage systems differiscono dagli altri sistemi basati su ONTAP (ASA, AFF e FAS) nell'implementazione del loro livello di storage.

NOTA	Solo il `ontap-nas` driver (con protocollo NFS) è supportato per i sistemi NetApp AFX; il protocollo SMB non è supportato.

Opzioni di configurazione del backend

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.

La tabella seguente mostra le opzioni di configurazione del backend:

Parametro	Descrizione	Predefinito
`version`		Sempre 1

Parametro	Descrizione	Predefinito
`storageDriverName`	Nome del driver di archiviazione **NOTA** Per i sistemi NetApp AFX, è supportato solo `ontap-nas`.	`ontap-nas`, `ontap-nas-economy`, o `ontap-nas-flexgroup`
`backendName`	Nome personalizzato o lo storage backend	Nome driver + "_" + dataLIF
`managementLIF`	Indirizzo IP di un cluster o di una LIF di gestione SVM. È possibile specificare un fully-qualified domain name (FQDN). Può essere impostato per usare indirizzi IPv6 se hai installato Trident usando il flag IPv6. Gli indirizzi IPv6 devono essere definiti tra parentesi quadre, ad esempio `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Per un passaggio senza interruzioni di MetroCluster, vedere Esempio di MetroCluster.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Indirizzo IP del protocollo LIF. NetApp consiglia di specificare `dataLIF`. Se non fornito, Trident recupera i dataLIF dall'SVM. Puoi specificare un fully-qualified domain name (FQDN) da usare per le operazioni di mount NFS, permettendoti di creare un DNS round-robin per bilanciare il carico tra più dataLIF. Può essere modificato dopo l'impostazione iniziale. Consulta . Può essere impostato per usare indirizzi IPv6 se hai installato Trident usando 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.** Vedere il Esempio di MetroCluster.	Indirizzo specificato o derivato da SVM, se non specificato (non consigliato)
`svm`	Macchina virtuale di storage da utilizzare **Ometti per MetroCluster.** Vedere il Esempio di MetroCluster.	Derivato se viene specificato un SVM `managementLIF`
`autoExportPolicy`	Abilita la creazione e l'aggiornamento automatici delle policy di esportazione [Boolean]. Utilizzando le `autoExportPolicy` e `autoExportCIDRs` opzioni, Trident può gestire automaticamente le policy di esportazione.	falso
`autoExportCIDRs`	Elenco di CIDR in base ai quali filtrare gli IP dei nodi Kubernetes quando `autoExportPolicy` è abilitato. Utilizzando le `autoExportPolicy` e `autoExportCIDRs` opzioni, Trident può gestire automaticamente le policy di esportazione.	["0.0.0.0/0", "::/0"]`
`labels`	Set di etichette arbitrarie in formato JSON da applicare ai volumi	""
`clientCertificate`	Valore codificato in Base64 del certificato client. Utilizzato per l'autenticazione basata su certificato	""

Parametro	Descrizione	Predefinito
`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. Facoltativo. Utilizzato per l'autenticazione basata su certificato	""
`username`	Nome utente per connettersi al cluster/SVM. Utilizzata per l'autenticazione basata su credenziali. Per l'autenticazione di Active Directory, vedere "Autenticare Trident a un backend SVM utilizzando le credenziali di Active Directory".	
`password`	Password per connettersi al cluster/SVM. Utilizzata per l'autenticazione basata su credenziali. Per l'autenticazione di Active Directory, vedere "Autenticare Trident a un backend SVM utilizzando le credenziali di Active Directory".	
`storagePrefix`	Prefisso utilizzato durante il provisioning di nuovi volumi nell'SVM. Non può essere aggiornato dopo averlo impostato **NOTA** Quando si utilizza ontap-nas-economy e un storagePrefix che è di 24 o più caratteri, i qtree non avranno il prefisso di archiviazione incorporato, anche se sarà presente nel nome del volume.	"Trident"

Parametro	Descrizione	Predefinito
`aggregate`	Aggregato per il provisioning (facoltativo; se impostato, deve essere assegnato alla SVM). Per il `ontap-nas-flexgroup` driver, questa opzione viene ignorata. Se non assegnato, qualsiasi degli aggregati disponibili può essere utilizzato per il provisioning di un FlexGroup volume. **NOTA** Quando l'aggregato viene aggiornato in SVM, viene aggiornato automaticamente in Trident interrogando 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 passerà allo stato di errore in Trident durante l'interrogazione dell'aggregato SVM. Devi modificare l'aggregato con uno presente sulla SVM o rimuoverlo completamente per riportare online il backend. **Non specificare per i sistemi di storage AFX.**	""
`limitAggregateUsage`	Il provisioning fallisce se l'utilizzo supera questa percentuale. **Non si applica ad Amazon FSx per ONTAP**. **Non specificare per i sistemi di storage 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 FlexGroup volume. Supportato per il driver di storage **ontap-nas-flexgroup**. **NOTA** Quando l'elenco degli aggregati viene aggiornato in SVM, l'elenco viene aggiornato automaticamente in Trident interrogando SVM senza dover riavviare il Trident Controller. Quando hai configurato un elenco di aggregati specifico in Trident per il provisioning dei volumi, se l'elenco degli aggregati viene rinominato o spostato fuori da SVM, il backend passerà allo stato di errore in Trident durante l'interrogazione dell'aggregato SVM. Devi modificare l'elenco degli aggregati con uno presente su SVM o rimuoverlo completamente per riportare online il backend.	""
`limitVolumeSize`	Il provisioning fallisce se la dimensione del volume richiesto supera questo valore.	"" (non applicato per impostazione predefinita)
`debugTraceFlags`	Flag di debug da usare per la risoluzione dei problemi. Ad esempio, {"api":false, "method":true} non utilizzare `debugTraceFlags` a meno che non si stia risolvendo un problema e si richieda un dump dettagliato del log.	null
`nasType`	Configura la creazione di volumi NFS o SMB. Le opzioni sono `nfs`, `smb` o null. Impostando su null, vengono creati di default volumi NFS. **Se specificato, impostare sempre su `nfs` per i sistemi di storage AFX**.	`nfs`
`nfsMountOptions`	Elenco separato da virgole delle opzioni di montaggio NFS. Le opzioni di montaggio per i volumi persistenti Kubernetes sono normalmente specificate nelle classi di storage, ma se non vengono specificate opzioni di montaggio in una classe di storage, Trident utilizzerà 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 imposterà alcuna opzione di montaggio su un volume persistente associato.	""
`qtreesPerFlexvol`	Numero massimo di Qtree per FlexVol, deve essere nell'intervallo [50, 300]	"200"

Parametro	Descrizione	Predefinito
`smbShare`	È possibile specificare uno dei seguenti valori: il nome di una condivisione SMB creata tramite Microsoft Management Console o ONTAP CLI; un nome per consentire a Trident di creare la condivisione SMB; oppure è possibile lasciare il parametro vuoto per impedire l'accesso condiviso ai volumi. Questo parametro è facoltativo per ONTAP on-premises. Questo parametro è obbligatorio per Amazon FSx for ONTAP backends e non può essere vuoto.	`smb-share`
`useREST`	Parametro booleano per utilizzare le API REST di ONTAP. `useREST` Quando impostato su `true`, Trident utilizza le ONTAP REST APIs per comunicare con il backend; quando impostato su `false`, Trident utilizza le chiamate ONTAPI (ZAPI) per comunicare con il backend. Questa funzione richiede ONTAP 9.11.1 e versioni successive. Inoltre, il ruolo di login ONTAP utilizzato deve avere accesso all'applicazione `ontapi`. Questo è soddisfatto dai ruoli predefiniti `vsadmin` e `cluster-admin`. A partire dalla release Trident 24.06 e ONTAP 9.15.1 o versioni successive, `useREST` è impostato su `true` per impostazione predefinita; modificare `useREST` su `false` per utilizzare le chiamate ONTAPI (ZAPI). **Se specificato, impostare sempre su `true` per i sistemi di storage AFX**.	`true` per ONTAP 9.15.1 o versioni successive, altrimenti `false`.
`limitVolumePoolSize`	Dimensione massima richiedibile FlexVol quando si utilizzano Qtrees nel backend ontap-nas-economy.	"" (non applicato per impostazione predefinita)
`denyNewVolumePools`	Restringe `ontap-nas-economy` i backend dal creare nuovi volumi FlexVol per contenere i loro Qtree. Solo i FlexVol preesistenti vengono utilizzati per il provisioning di nuovi PV.	
`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 del backend per il provisioning dei volumi

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

Parametro	Descrizione	Predefinito
`spaceAllocation`	Allocazione dello spazio per Qtrees	"true"
`spaceReserve`	Modalità di prenotazione dello spazio; "none" (thin) o "volume" (thick)	"none"

Parametro	Descrizione	Predefinito
`snapshotPolicy`	policy di Snapshot da utilizzare	"none"
`qosPolicy`	Gruppo di policy QoS da assegnare ai volumi creati. Scegliere uno tra qosPolicy o adaptiveQosPolicy per storage pool/backend	""
`adaptiveQosPolicy`	Gruppo di policy QoS adattivo da assegnare ai volumi creati. Scegliere uno tra qosPolicy o adaptiveQosPolicy per pool di storage/backend. Non supportato da ontap-nas-economy.	""
`snapshotReserve`	Percentuale di volume riservata alle snapshot	"0" se `snapshotPolicy` è "none", altrimenti ""
`splitOnClone`	Dividere un clone dal suo genitore al momento della creazione	"false"
`encryption`	Abilita 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 fornito in Trident sarà abilitato per NAE. Per ulteriori informazioni, fare riferimento a: "Come funziona Trident con NVE e NAE".	"false"
`tieringPolicy`	Policy di tiering da utilizzare "none"	
`unixPermissions`	Modalità per nuovi volumi	"777" per i volumi NFS; vuoto (non applicabile) per i volumi SMB
`snapshotDir`	Controlla l'accesso alla `.snapshot` directory	`true`, `false` (Impostato esplicitamente).
`exportPolicy`	Policy di esportazione da utilizzare	"default"
`securityStyle`	Stile di sicurezza per i nuovi volumi. NFS supporta `mixed` e `unix` stili di sicurezza. SMB supporta `mixed` e `ntfs` stili di sicurezza.	L'impostazione predefinita di NFS è `unix`. L'impostazione predefinita di SMB è `ntfs`.
`nameTemplate`	Modello per creare nomi di volume personalizzati.	""

NOTA L'utilizzo di gruppi di policy QoS con Trident richiede ONTAP 9.8 o versioni successive. Dovresti utilizzare un gruppo di policy QoS non condiviso e assicurarti che il gruppo di policy venga applicato a ciascun componente individualmente. Un gruppo di policy QoS condiviso impone il limite massimo per il throughput di tutti i carichi di lavoro.

Esempi di provisioning del volume

Ecco un esempio con i valori predefiniti:

```
---
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 il FlexVol sia dimensionato correttamente con la percentuale di snapshotReserve e il 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 v21.07, quando l'utente richiedeva un PVC (ad esempio, 5 GiB), con la snapshotReserve al 50 percento, otteneva solo 2,5 GiB di spazio scrivibile. Questo perché ciò che l'utente richiedeva era l'intero volume e `snapshotReserve` era una percentuale di quello. Con Trident 21.07, ciò che l'utente richiede è lo spazio scrivibile e Trident definisce il numero di `snapshotReserve` come percentuale dell'intero volume. Questo non si applica a `ontap-nas-economy`. Vedi il seguente esempio per capire come funziona:

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 5/0,5 = 10 GiB e la dimensione disponibile è 5 GiB, che è quanto richiesto dall'utente nella richiesta PVC. Il comando `volume show` dovrebbe mostrare risultati simili a questo esempio:

I backend esistenti delle installazioni precedenti eseguiranno il provisioning dei volumi come spiegato sopra durante l'aggiornamento Trident. Per i volumi che hai creato prima dell'aggiornamento, dovresti ridimensionare i loro 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, si forniscono all'applicazione 3 GiB di spazio scrivibile su un volume da 6 GiB.

Esempi di configurazione minima

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.

NOTA	Se si utilizza Amazon FSx su NetApp ONTAP con Trident, si consiglia di specificare i nomi DNS per i 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 ONTAP NAS FlexGroup

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

Esempio di 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 SVM".

Per un passaggio e un ritorno senza interruzioni, specificare l'SVM utilizzando `managementLIF` e omettere i parametri `dataLIF` e `svm`. 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

Questo è un esempio di configurazione minima del backend. `clientCertificate`, `clientPrivateKey`, e `trustedCACertificate` (facoltativo, se si utilizza una CA attendibile) vengono popolati in `backend.json` e accettano rispettivamente i valori codificati in base64 del certificato client, della chiave privata e del certificato CA attendibile.

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

Esempio di policy di esportazione automatica

Questo esempio mostra come istruire Trident a utilizzare policy di esportazione dinamiche per creare e gestire automaticamente la policy di esportazione. Questo funziona allo stesso modo per i driver `ontap-nas-economy` 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 indirizzi IPv6

Questo esempio mostra `managementLIF` l'utilizzo di 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 for ONTAP che utilizza volumi SMB

Il `smbShare` parametro è obbligatorio per Amazon FSx for 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 del 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 del backend di esempio mostrati di seguito, vengono impostati valori predefiniti specifici per tutti i pool di storage, come `spaceReserve` a none, `spaceAllocation` a false e `encryption` a false. I pool virtuali sono definiti nella sezione storage.

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

In questi esempi, alcuni pool di storage impostano i propri `spaceReserve`, `spaceAllocation` e `encryption` valori, mentre alcuni pool sovrascrivono i valori predefiniti.

Esempio 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 ONTAP NAS FlexGroup

```
---
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"
```

Mappa i backend a 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 avrà gli aspetti definiti nel pool virtuale scelto.

- Il `protection-gold` StorageClass verrà mappato sul primo e sul secondo pool virtuale nel `ontap-nas-flexgroup` backend. Questi sono gli unici pool che offrono protezione di livello gold.

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

- Il `protection-not-gold` StorageClass verrà mappato sul terzo e 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"
```

- Il `app-mysqldb` StorageClass verrà mappato sul 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"
```

- Il `protection-silver-creditpoints-20k` StorageClass verrà mappato 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"
```

- Il `creditpoints-5k` StorageClass verrà mappato sul terzo pool virtuale nel `ontap-nas` backend e sul 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 deciderà quale pool virtuale selezionare e garantirà che il requisito di storage sia soddisfatto.

Aggiorna `dataLIF` dopo la configurazione iniziale

È possibile modificare il dataLIF dopo la configurazione iniziale eseguendo il comando seguente per fornire il nuovo file JSON backend con il dataLIF aggiornato.

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

NOTA Se i PVC sono collegati a uno o più pod, è necessario disattivare tutti i pod corrispondenti e quindi riattivarli affinché il nuovo dataLIF abbia effetto.

Esempi di SMB sicuro

Configurazione del 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 storage

```
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 storage class 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 le annotazioni, indipendentemente dalle configurazioni impostate nel Backend o nel PVC.

Esempio di storage class 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
```

Amazon FSx for NetApp ONTAP

Usa Trident con Amazon FSx for NetApp ONTAP

"Amazon FSx for 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 di Trident", per integrare FSx for ONTAP con Trident, è necessario:

- Un cluster Amazon EKS esistente o un cluster Kubernetes autogestito con `kubectl` installato.
- Un file system Amazon FSx for NetApp ONTAP esistente e una macchina virtuale di storage (SVM) che sia raggiungibile dai nodi worker del cluster.
- Nodi worker che sono preparati per "NFS o iSCSI".

> **NOTA** Assicurati di seguire i passaggi di preparazione del nodo richiesti per Amazon Linux e Ubuntu "Amazon Machine Images" (AMIs) a seconda del tipo di EKS AMI.

Considerazioni

- Volumi SMB:
  - I volumi SMB sono supportati utilizzando solo il `ontap-nas` driver.
  - I volumi SMB non sono supportati con il componente aggiuntivo Trident EKS.
  - Trident supporta volumi SMB montati solo su pod in esecuzione su nodi Windows. Consultare "Prepararsi al provisioning dei volumi SMB" per i dettagli.
- Prima di Trident 24.02, i volumi creati su Amazon FSx file system che hanno backup automatici abilitati non potevano essere eliminati da Trident. Per evitare questo problema in Trident 24.02 o versioni successive, specificare l' `fsxFilesystemID`, AWS `apiRegion`, AWS `apikey` e AWS `secretKey` nel file di configurazione backend per AWS FSx for ONTAP.

> **NOTA** Se si specifica un ruolo IAM per Trident, è possibile omettere di specificare i campi `apiRegion`, `apiKey` e `secretKey` a Trident esplicitamente. Per ulteriori informazioni, fai riferimento a "Opzioni ed esempi di configurazione di FSx per ONTAP".

Utilizzo simultaneo di Trident SAN/iSCSI e del driver EBS-CSI

Se prevedi di utilizzare 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 l' `vsadmin`utente configurato per la tua SVM.

ATTENZIONE	Trident prevede di essere eseguito come `vsadmin` utente SVM o come utente con un nome diverso che abbia lo stesso ruolo. Amazon FSx for NetApp ONTAP ha un `fsxadmin` utente che è una sostituzione limitata dell'utente `admin` cluster di ONTAP. Si consiglia vivamente di utilizzare `vsadmin` con Trident.

- Basato su certificato: Trident comunicherà con l'SVM sul tuo file system FSx utilizzando un certificato installato sul tuo SVM.

Per i dettagli sull'abilitazione dell'autenticazione, fare riferimento all'autenticazione per il tipo di driver:

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

Amazon Machine Images (AMI) testate

EKS cluster supporta vari sistemi operativi, ma AWS ha ottimizzato alcune Amazon Machine Images (AMIs) per container ed EKS. Le seguenti AMI sono state testate con NetApp Trident 25.02.

AMI	NAS	NAS-economy	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 l'AMI desiderata non è elencata qui, non significa che non sia supportata; significa semplicemente che non è stata testata. Questo elenco serve come guida per le AMI di cui è noto il funzionamento.

Test eseguiti con:

- EKS versione: 1.32
- Metodo di installazione: Helm 25.06 e come AWS add-On 25.06
- Per NAS sono stati testati sia NFSv3 che NFSv4.1.
- Per SAN è stato testato solo iSCSI, non NVMe-oF.

Test eseguiti:

- Crea: Storage Class, pvc, pod
- Elimina: pod, pvc (normale, qtree/lun – economy, NAS con backup AWS)

Trova ulteriori informazioni

- "Documentazione di Amazon FSx for NetApp ONTAP"
- "Post del blog su Amazon FSx for NetApp ONTAP"

Crea un ruolo IAM e un AWS Secret

È possibile configurare i pod Kubernetes per accedere alle risorse AWS autenticandosi come ruolo AWS IAM invece di fornire credenziali AWS esplicite.

NOTA	Per eseguire l'autenticazione tramite un ruolo AWS IAM, è necessario disporre di un cluster Kubernetes distribuito tramite EKS.

Crea un secret di AWS Secrets Manager

Poiché Trident emetterà API contro un FSx vserver per gestire lo storage per te, avrà bisogno di credenziali per farlo. Il modo sicuro per trasmettere tali credenziali è tramite un segreto AWS Secrets Manager. Pertanto, se non ne hai già uno, dovrai creare un segreto AWS Secrets Manager che contenga le credenziali per l'account vsadmin.

Questo esempio crea un segreto AWS Secrets Manager per archiviare le credenziali Trident CSI:

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

Crea policy IAM

Anche Trident necessita delle autorizzazioni AWS per funzionare correttamente. Pertanto, è necessario creare una policy che dia a Trident le autorizzazioni di cui ha bisogno.

I seguenti esempi creano una policy IAM utilizzando l'AWS CLI:

```
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"
```

Esempio di policy JSON:

```
{
  "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 Pod Identity o 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 IAM role for Service account association (IRSA). Tutti i Pod configurati per utilizzare l'account di servizio possono quindi accedere a qualsiasi servizio AWS a cui il ruolo ha permessi di accesso.

Identità del pod

Le associazioni Amazon EKS Pod Identity consentono di gestire le credenziali per le tue applicazioni, in modo simile a come i profili delle istanze Amazon EC2 forniscono 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 ulteriori informazioni, fare riferimento a "Configura 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 Pod Identity. Quindi crea un ruolo con questa trust policy:

```
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 che è stato 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 il service account 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-relationship.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"
        }
      }
    }
  ]
}
```

Aggiorna i seguenti valori nel file `trust-relationship.json`:

- **<account_id>** - ID del tuo account AWS
- **<oidc_provider>** - L'OIDC del tuo cluster EKS. Puoi ottenere l'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, associare il criterio (che è stato creato nel passaggio sopra) al ruolo utilizzando questo comando:

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

Verifica che il provider OICD sia associato:

Verifica che il tuo provider OIDC sia associato al tuo cluster. Puoi verificarlo utilizzando questo 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 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:

- Helm
- Componente aggiuntivo EKS

Se desideri utilizzare la funzionalità di snapshot, installa il componente aggiuntivo CSI snapshot controller. Consulta "Abilita la funzionalità snapshot per i volumi CSI" per maggiori informazioni.

Installa Trident tramite helm

Identità del pod

1. Aggiungi 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
```

Puoi utilizzare il comando `helm list` per rivedere i dettagli dell'installazione come nome, namespace, chart, 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 account di servizio (IRSA)

1. Aggiungi 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
```

Puoi utilizzare il comando `helm list` per rivedere i dettagli dell'installazione come nome, namespace, chart, 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 si prevede di utilizzare iSCSI, assicurarsi che iSCSI sia abilitato sul computer client. Se si utilizza il sistema operativo AL2023 Worker node, è possibile 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}
```

Installa Trident tramite il componente aggiuntivo EKS

Il componente aggiuntivo Trident EKS include le patch di sicurezza più recenti, le correzioni di bug ed è convalidato da AWS per funzionare con Amazon EKS. Il componente aggiuntivo EKS consente di garantire costantemente che i cluster Amazon EKS siano sicuri e stabili e di ridurre la quantità di lavoro necessaria per installare, configurare e aggiornare i componenti aggiuntivi.

Prerequisiti

Assicurati di avere quanto segue prima di configurare il componente aggiuntivo Trident per AWS EKS:

- Un account cluster Amazon EKS con abbonamento aggiuntivo
- Autorizzazioni AWS per l'AWS marketplace:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Tipo 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

Abilita il componente aggiuntivo Trident per AWS

Console di gestione

1. Apri la console Amazon EKS su https://console.aws.amazon.com/eks/home#/clusters.
2. Nel riquadro di navigazione a sinistra, seleziona **Clusters**.
3. Seleziona il nome del cluster per cui desideri configurare il componente aggiuntivo NetApp Trident CSI.
4. Seleziona **Componenti aggiuntivi** e poi seleziona **Ottieni altri componenti aggiuntivi**.
5. Seguire questi passaggi per selezionare il software add-on:
   a. Scorri verso il basso fino alla sezione **AWS Marketplace add-ons** e digita **"Trident"** nella casella di ricerca.
   b. Selezionare la check box nell'angolo in alto a destra della casella Trident by NetApp.
   c. Seleziona **Next**.
6. Nella pagina delle impostazioni **Configura i componenti aggiuntivi selezionati**, eseguire le seguenti operazioni:

   NOTA **Salta questi passaggi se utilizzi l'associazione Pod Identity.**

   a. Seleziona la **Version** che desideri utilizzare.
   b. Se si utilizza l'autenticazione IRSA, assicurarsi di impostare i valori di configurazione disponibili nelle impostazioni di configurazione opzionali:
      - Seleziona la **Version** che desideri utilizzare.
      - Seguire lo **Schema di configurazione aggiuntivo** e impostare il parametro **configurationValues** nella sezione **Valori di configurazione** sul role-arn creato nel passaggio precedente (il valore deve essere nel formato seguente):

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

   +
   Se si seleziona Override per il metodo di risoluzione dei conflitti, una o più impostazioni del componente aggiuntivo esistente possono essere sovrascritte con le impostazioni dell'add-on Amazon EKS. Se non si abilita 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, assicurarsi che l'add-on Amazon EKS non gestisca impostazioni che è necessario autogestire.
7. Scegli **Next**.
8. Nella pagina **Revisione e aggiunta**, scegliere **Crea**.

   Al termine dell'installazione del software add-on, viene visualizzato il software add-on installato.

AWS CLI

1. Crea il `add-on.json` file:

Per Pod Identity, 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 Sostituisci `<role ARN>` con l'ARN del ruolo che è stato creato nel passaggio precedente.

2. Installare il Trident EKS add-on.

```
aws eks create-addon --cli-input-json file://add-on.json
```

eksctl

Il seguente esempio di comando installa il Trident EKS add-on:

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

Aggiornare il software add-on Trident EKS

Console di gestione

1. Apri la console Amazon EKS https://console.aws.amazon.com/eks/home#/clusters.
2. Nel riquadro di navigazione a sinistra, seleziona **Clusters**.
3. Selezionare il nome del cluster per cui si desidera aggiornare il software add-on NetApp Trident CSI.
4. Selezionare la scheda **Add-ons**.
5. Seleziona **Trident by NetApp** e poi seleziona **Modifica**.
6. Nella pagina **Configura Trident by NetApp**, procedere come segue:
   a. Seleziona la **Version** che desideri utilizzare.
   b. Espandi le **Impostazioni di configurazione opzionali** e modificale secondo necessità.
   c. Seleziona **Salva modifiche**.

AWS CLI

Il seguente esempio aggiorna l'add-on 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

- Controlla la versione corrente del tuo software add-on FSxN Trident CSI. Sostituisci `my-cluster` con il nome del tuo 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 software add-on alla versione riportata sotto UPDATE AVAILABLE 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 qualsiasi delle impostazioni Amazon EKS add-on è in conflitto con le impostazioni esistenti, l'aggiornamento dell'Amazon EKS add-on non riesce; viene visualizzato un messaggio di errore per aiutarti a risolvere il conflitto. Prima di specificare questa opzione, assicurati che l'Amazon EKS add-on non gestisca impostazioni che devi gestire, perché tali impostazioni vengono sovrascritte con questa opzione. Per ulteriori informazioni su altre opzioni per questa impostazione, vedi "Componenti aggiuntivi". Per ulteriori informazioni sulla gestione dei campi Amazon EKS Kubernetes, vedi "Gestione dei campi Kubernetes".

Disinstallare/rimuovere il Trident EKS add-on

Hai due opzioni per rimuovere un add-on di Amazon EKS:

- **Conserva il software add-on sul tuo cluster** – Questa opzione rimuove la gestione di qualsiasi impostazione da parte di Amazon EKS. Rimuove anche la possibilità per Amazon EKS di notificarti gli aggiornamenti e di aggiornare automaticamente l'add-on Amazon EKS dopo che hai avviato un aggiornamento. Tuttavia, conserva il software add-on sul tuo cluster. Questa opzione rende l'add-on un'installazione autogestita, invece che un add-on Amazon EKS. Con questa opzione, non c'è alcun downtime per l'add-on. Mantieni l' `--preserve` opzione nel comando per conservare l'add-on.
- **Rimuovere il software add-on interamente dal cluster** – NetApp consiglia di rimuovere l'add-on Amazon EKS dal cluster solo se non ci sono risorse sul cluster che dipendono da esso. Rimuovere l'opzione `--preserve` dal comando `delete` per rimuovere l'add-on.

NOTA Se al software add-on è associato un account IAM, l'account IAM non viene rimosso.

Console di gestione

1. Apri la console Amazon EKS su https://console.aws.amazon.com/eks/home#/clusters.
2. Nel riquadro di navigazione sinistro, selezionare **Clusters**.
3. Selezionare il nome del cluster dal quale si desidera rimuovere il software add-on NetApp Trident CSI.
4. Selezionare la scheda **Add-ons** e poi scegliere **Trident by NetApp**.*
5. Seleziona **Rimuovi**.
6. Nella finestra di dialogo **Remove netapp_trident-operator confirmation**, procedere come segue:
   a. Se si desidera che Amazon EKS smetta di gestire le impostazioni del software add-on, selezionare **Preserva sul cluster**. Eseguire questa operazione se si desidera conservare il software add-on sul cluster in modo da poter gestire autonomamente tutte le impostazioni del software add-on.
   b. Immettere **netapp_trident-operator**.
   c. Seleziona **Rimuovi**.

AWS CLI

Sostituire `my-cluster` con il nome del cluster, quindi eseguire 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	Richiesto	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

```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

```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 di storage **basic-csi** sia in Kubernetes che in Trident, e Trident dovrebbe aver rilevato i pool sul backend.

```
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

È possibile integrare Trident con Amazon FSx for 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 i dettagli sui driver, fare riferimento a "Driver NAS" e "Driver SAN".

Crea il backend

Dopo aver creato il file di configurazione, eseguire i seguenti comandi per creare e convalidare la Trident Backend Configuration (TBC):

- Crea la configurazione del backend Trident (TBC) dal file yaml ed esegui il seguente comando:

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- Verificare che la configurazione del backend 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 [Backend-advanced-configuration-and-examples] 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 storage 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, aggiungete il necessario `mountOptions` alla storage class:

```
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 effettuare il provisioning dei volumi di un determinato tipo.

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

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

[1]: Non supportato da ONTAP Select o FSx for ONTAP

Fate riferimento a "Oggetti Kubernetes e Trident" per i dettagli su come le classi di storage interagiscono con `PersistentVolumeClaim` e sui parametri per controllare come Trident effettua il provisioning dei volumi.

Crea la classe di archiviazione

Una volta configurato il StorageClass, puoi crearlo in Kubernetes.

Passaggi

1. Si tratta di un oggetto Kubernetes, quindi usa `kubectl` per crearlo in Kubernetes.

```
kubectl create -f storage-class-ontapnas.yaml
```

2. Ora dovresti vedere una classe di storage **basic-csi** sia in Kubernetes che in Trident, e Trident dovrebbe aver rilevato i pool sul backend.

```
kubectl get sc basic-csi
```

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

Effettuare il provisioning dei volumi SMB

È possibile eseguire il provisioning dei volumi SMB utilizzando il `ontap-nas` driver. Tuttavia, per farlo è necessario completare questi passaggi: "Prepararsi al provisioning dei volumi SMB".

Configurazione avanzata del backend ed esempi

Consulta la tabella seguente per le opzioni di configurazione del backend:

Parametro	Descrizione	Esempio
`version`		Sempre 1

Parametro	Descrizione	Esempio
`storageDriverName`	Nome del driver di archiviazione	`ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`
`backendName`	Nome personalizzato o lo storage backend	Nome driver + "_" + dataLIF
`managementLIF`	Indirizzo IP di un cluster o di una LIF di gestione SVM. È possibile specificare un fully-qualified domain name (FQDN). Può essere impostato per usare indirizzi IPv6 se hai installato Trident usando il flag IPv6. Gli indirizzi IPv6 devono essere definiti tra parentesi quadre, ad esempio [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Se si fornisce `fsxFilesystemID` sotto il campo `aws`, non è necessario fornire `managementLIF` perché Trident recupera le informazioni SVM `managementLIF` da AWS. Quindi, è necessario fornire le credenziali per un utente sotto l'SVM (ad esempio: vsadmin) e l'utente deve avere il ruolo `vsadmin`.	"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. Puoi specificare un fully-qualified domain name (FQDN) da usare per le operazioni di mount NFS, permettendoti di creare un DNS round-robin per bilanciare il carico tra più dataLIF. Può essere modificato dopo l'impostazione iniziale. **ONTAP SAN drivers**: non specificare per iSCSI. Trident usa ONTAP Selective LUN Map per individuare i LIF iSCSI necessari a stabilire una sessione multipath. Viene generato un avviso se dataLIF è definito esplicitamente. Può essere impostato per usare indirizzi IPv6 se hai installato Trident usando 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 delle policy di esportazione [Boolean]. Utilizzando le `autoExportPolicy` e `autoExportCIDRs` opzioni, Trident può gestire automaticamente le policy di esportazione.	`false`
`autoExportCIDRs`	Elenco di CIDR in base ai quali filtrare gli IP dei nodi Kubernetes quando `autoExportPolicy` è abilitato. Utilizzando le `autoExportPolicy` e `autoExportCIDRs` opzioni, Trident può gestire automaticamente le policy di esportazione.	"["0.0.0.0/0", "::/0"]"
`labels`	Set di etichette arbitrarie in formato JSON da applicare ai volumi	""
`clientCertificate`	Valore codificato in Base64 del certificato client. 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. Facoltativo. Utilizzato per l'autenticazione basata su certificato.	""
`username`	Nome utente per connettersi al cluster o alla SVM. Utilizzata per l'autenticazione basata sulle credenziali. Ad esempio, vsadmin.	
`password`	Password per connettersi al cluster o alla SVM. Utilizzata per l'autenticazione basata sulle credenziali.	
`svm`	Macchina virtuale di storage da utilizzare	Derivato se viene specificato un SVM managementLIF.
`storagePrefix`	Prefisso utilizzato durante il provisioning di nuovi volumi nell'SVM. Non può essere modificato dopo la creazione. Per aggiornare questo parametro, sarà necessario creare un nuovo backend.	`trident`
`limitAggregateUsage`	**Non specificare per Amazon FSx per NetApp ONTAP.** I `fsxadmin` e `vsadmin` forniti non contengono le autorizzazioni necessarie per recuperare l'utilizzo aggregato e limitarlo tramite Trident.	Non utilizzare.
`limitVolumeSize`	Il provisioning fallisce se la dimensione del volume richiesto supera questo valore. Limita inoltre la dimensione massima dei volumi che gestisce 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`	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 usare per la risoluzione dei problemi. Ad esempio, {"api":false, "method":true} non utilizzare `debugTraceFlags` a meno che non si stia risolvendo un problema e si richieda un dump dettagliato del log.	null
`nfsMountOptions`	Elenco separato da virgole delle opzioni di montaggio NFS. Le opzioni di montaggio per i volumi persistenti Kubernetes sono normalmente specificate nelle classi di storage, ma se non vengono specificate opzioni di montaggio in una classe di storage, Trident utilizzerà 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 imposterà alcuna opzione di montaggio su un volume persistente associato.	""
`nasType`	Configura la creazione di volumi NFS o SMB. Le opzioni sono `nfs`, `smb` o null. **Deve essere impostato su `smb` per i volumi SMB.** Impostando su null, vengono creati di default volumi NFS.	`nfs`
`qtreesPerFlexvol`	Numero massimo di qtree per volume FlexVol, deve essere compreso nell'intervallo [50, 300]	`"200"`
`smbShare`	È possibile specificare uno dei seguenti parametri: il nome di una condivisione SMB creata tramite Microsoft Management Console o ONTAP CLI oppure un nome che consenta a Trident di creare la condivisione SMB. Questo parametro è obbligatorio per Amazon FSx for ONTAP backends.	`smb-share`

Parametro	Descrizione	Esempio
`useREST`	Parametro booleano per utilizzare le API REST di ONTAP. Se impostato su `true`, Trident utilizzerà le API REST di ONTAP per comunicare con il backend. Questa funzione richiede ONTAP 9.11.1 e versioni successive. Inoltre, il ruolo di login ONTAP utilizzato deve avere accesso all'applicazione `ontap`. Questo è soddisfatto dai ruoli predefiniti `vsadmin` e `cluster-admin`.	`false`
`aws`	È possibile specificare quanto segue nel file di configurazione per AWS FSx for ONTAP: - `fsxFilesystemID`: Specificare l'ID del file system AWS FSx. - `apiRegion`: Nome della regione API AWS. - `apikey`: Chiave API AWS. - `secretKey`: Chiave segreta AWS.	`""` `""` `""`
`credentials`	Specificare le credenziali FSx SVM da archiviare in AWS Secrets Manager. - `name`: Amazon Resource Name (ARN) del segreto, che contiene le credenziali di SVM. - `type`: Impostare su `awsarn`. Consulta "Crea un segreto AWS Secrets Manager" per maggiori informazioni.	

Opzioni di configurazione del backend per il provisioning dei volumi

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

Parametro	Descrizione	Predefinito
`spaceAllocation`	Allocazione dello spazio per le LUN	`true`
`spaceReserve`	Modalità di prenotazione dello spazio; "none" (thin) o "volume" (thick)	`none`
`snapshotPolicy`	policy di Snapshot da utilizzare	`none`

Parametro	Descrizione	Predefinito
`qosPolicy`	Gruppo di policy QoS da assegnare ai volumi creati. Scegli uno tra qosPolicy o adaptiveQosPolicy per ogni pool di storage o backend. L'utilizzo di gruppi di policy QoS con Trident richiede ONTAP 9.8 o versioni successive. È consigliabile utilizzare un gruppo di policy QoS non condiviso e assicurarsi che il gruppo di policy venga applicato a ciascun componente singolarmente. Un gruppo di policy QoS condiviso impone il limite massimo per il throughput di tutti i carichi di lavoro.	""
`adaptiveQosPolicy`	Gruppo di policy QoS adattivo da assegnare ai volumi creati. Scegli uno tra qosPolicy o adaptiveQosPolicy per ogni pool di storage o backend. Non supportato da ontap-nas-economy.	""
`snapshotReserve`	Percentuale di volume riservata per gli snapshot "0"	Se `snapshotPolicy` è `none`, `else` ""
`splitOnClone`	Dividere un clone dal suo genitore al momento della creazione	`false`
`encryption`	Abilita 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 fornito in Trident sarà abilitato per NAE. Per ulteriori informazioni, fare riferimento a: "Come funziona Trident con NVE e NAE".	`false`
`luksEncryption`	Abilita la crittografia LUKS. Consulta "Usa Linux Unified Key Setup (LUKS)". Solo SAN.	""
`tieringPolicy`	Criterio di tiering da utilizzare `none`	
`unixPermissions`	Modalità per nuovi volumi. **Lasciare vuoto per volumi SMB.**	""
`securityStyle`	Stile di sicurezza per i nuovi volumi. NFS supporta `mixed` e `unix` stili di sicurezza. SMB supporta `mixed` e `ntfs` stili di sicurezza.	L'impostazione predefinita di NFS è `unix`. L'impostazione predefinita 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.

Crea 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.

Esempi di manifest

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
```

Esempio di PVC utilizzando iSCSI

Questo esempio mostra un PVC di base per iSCSI con accesso RWO che è associato a un StorageClass denominato `protection-gold`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: protection-gold
```

Crea PVC

Passaggi

1. Crea 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
```

Fate riferimento a "Oggetti Kubernetes e Trident" per i dettagli su come le classi di storage interagiscono con `PersistentVolumeClaim` e sui parametri per controllare come Trident effettua il provisioning dei volumi.

Distribuisci un'applicazione

Una volta creati la classe di storage e il PVC, è possibile montare il PV su un pod. Questa sezione elenca il comando di esempio e la configurazione per collegare il PV a un pod.

Distribuisci un'applicazione di esempio

Passaggi

1. Monta il volume in un pod.

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

Questi esempi mostrano 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 Puoi monitorare l'avanzamento usando `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
```

Ora puoi eliminare il Pod. L'applicazione Pod non esisterà più, ma il volume rimarrà.

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

Configura il componente aggiuntivo Trident EKS su un cluster EKS

NetApp Trident semplifica la gestione dello storage Amazon FSx for NetApp ONTAP in Kubernetes, consentendo a sviluppatori e amministratori di concentrarsi sulla distribuzione delle applicazioni. Il componente aggiuntivo NetApp Trident EKS include le patch di sicurezza più recenti, le correzioni di bug ed è convalidato da AWS per funzionare con Amazon EKS. Il componente aggiuntivo EKS consente di garantire

costantemente che i cluster Amazon EKS siano sicuri e stabili e di ridurre la quantità di lavoro necessaria per installare, configurare e aggiornare i componenti aggiuntivi.

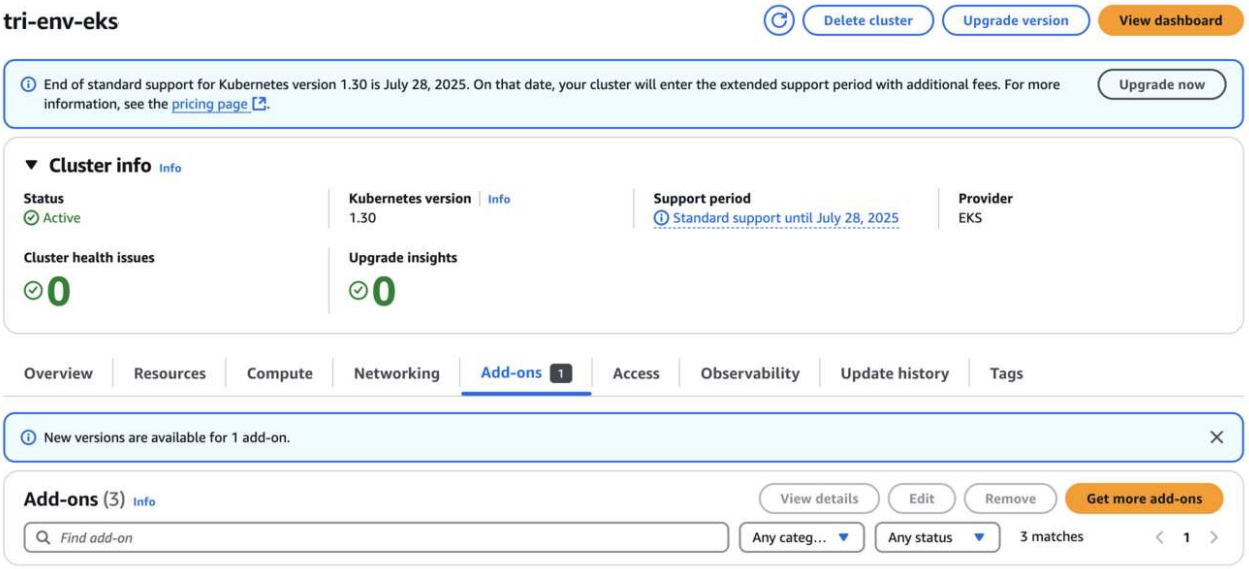
Prerequisiti

Assicurati di avere quanto segue prima di configurare il componente aggiuntivo Trident per AWS EKS:

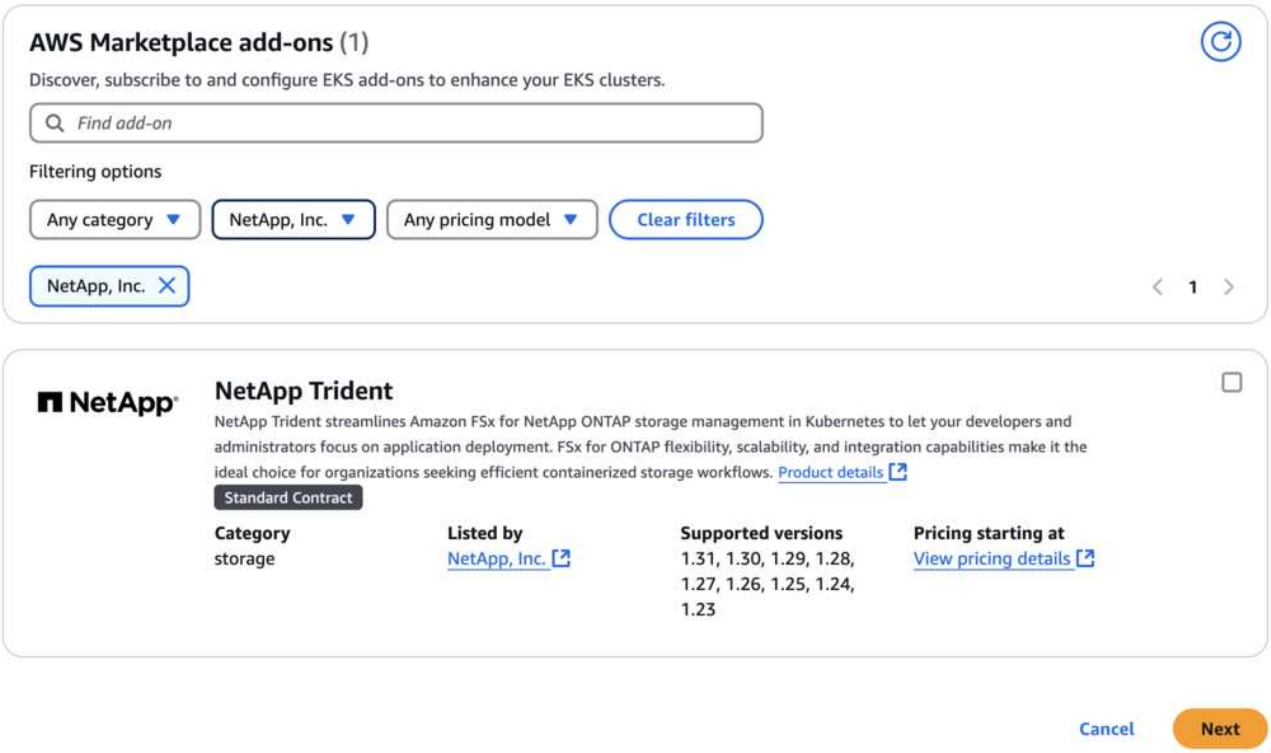
- Un account cluster Amazon EKS con autorizzazioni per utilizzare i componenti aggiuntivi. Consulta "Componenti aggiuntivi Amazon EKS".
- Autorizzazioni AWS per l'AWS marketplace:
  ```
  "aws-marketplace:ViewSubscriptions",
  "aws-marketplace:Subscribe",
  "aws-marketplace:Unsubscribe
  ```
- Tipo AMI: 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

Passaggi

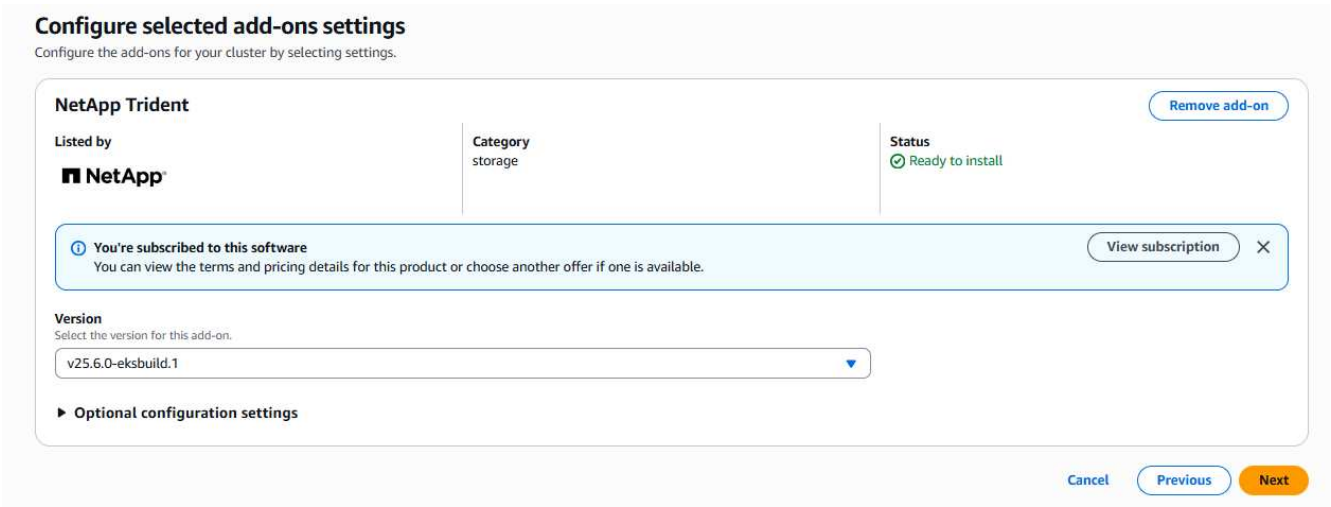
1. Assicurati di creare il ruolo IAM e il segreto AWS per consentire ai pod EKS di accedere alle risorse AWS. Per istruzioni, consulta "Crea un ruolo IAM e un AWS Secret".
2. Nel tuo cluster EKS Kubernetes, vai alla scheda **Componenti aggiuntivi**.



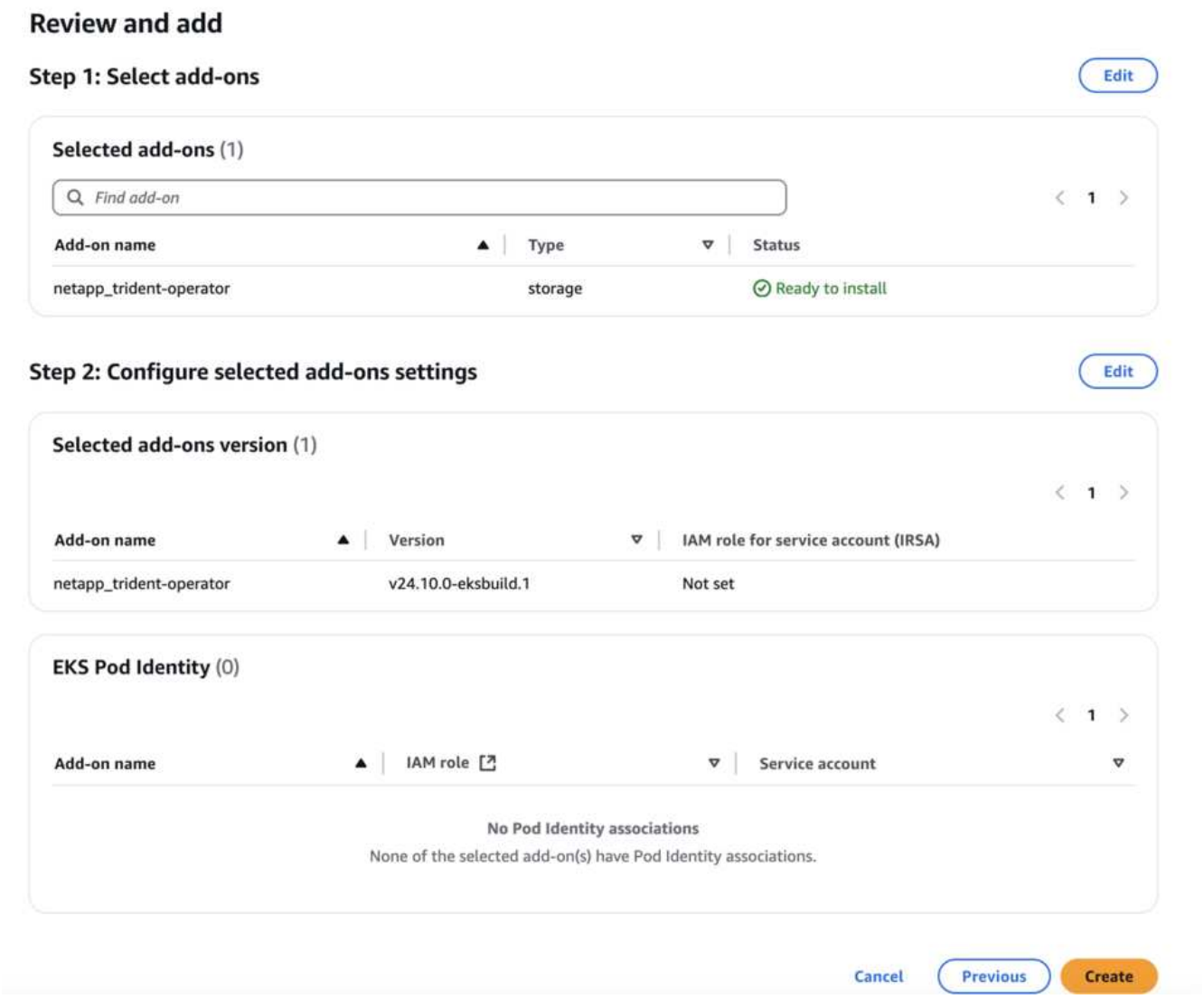
3. Vai su **AWS Marketplace add-ons** e scegli la categoria *storage*.



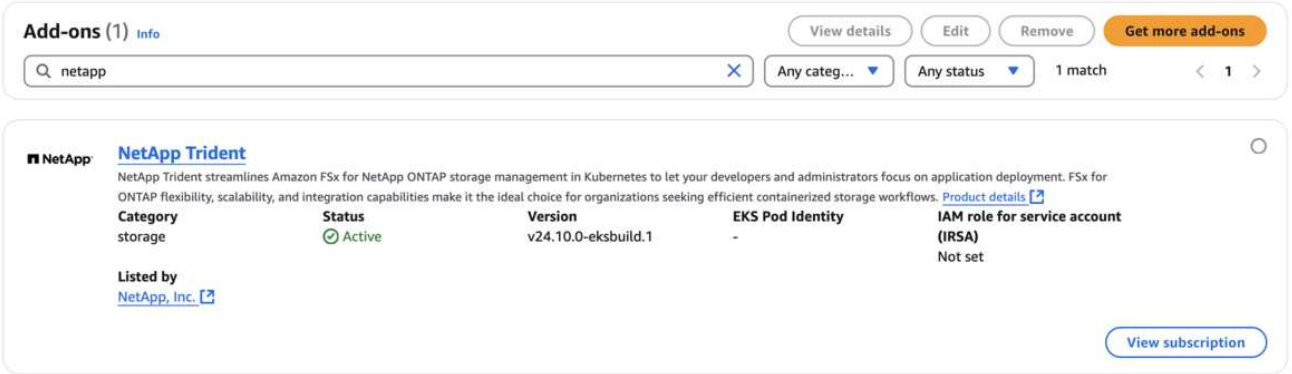
4. Individua **NetApp Trident** e seleziona la casella di controllo per il componente aggiuntivo Trident, quindi fai clic su **Avanti**.
5. Scegli la versione desiderata dell'add-on.



6. Configura le impostazioni aggiuntive richieste.



7. Se si utilizza IRSA (IAM roles for service account), fare riferimento ai passaggi di configurazione aggiuntivi "qui".
8. Seleziona **Create**.
9. Verificare che lo stato del componente aggiuntivo sia *Active*.



10. Eseguire il seguente comando per verificare che Trident sia installato correttamente sul cluster:

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

11. Continua la configurazione e configura il backend di storage. Per informazioni, vedi "Configura il backend di storage".

Installa/disinstalla il componente aggiuntivo Trident EKS tramite CLI

Installa il componente aggiuntivo Trident EKS di NetApp 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)

Disinstalla il componente aggiuntivo NetApp Trident EKS utilizzando la 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 storage. Indica a Trident come comunicare con quel sistema storage e come Trident deve eseguire il provisioning dei volumi da esso. Dopo l'installazione di Trident, il passo successivo è creare un backend. La `TridentBackendConfig` Custom Resource Definition (CRD) consente di creare e gestire i backend di Trident direttamente attraverso l'interfaccia di Kubernetes. Puoi farlo utilizzando `kubectl` o lo strumento CLI equivalente per la tua distribuzione Kubernetes.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`) è un CRD frontend e namespaced che consente di gestire i backend Trident utilizzando `kubectl`. Gli amministratori di Kubernetes e dello storage possono ora creare e gestire i backend direttamente attraverso la CLI di Kubernetes senza richiedere un'utility a riga di comando dedicata (`tridentctl`).

Alla creazione di un `TridentBackendConfig` oggetto, avviene quanto segue:

- Un backend viene creato automaticamente da Trident in base alla configurazione che fornisci. Questo è rappresentato internamente come un `TridentBackend` (`tbe`, `tridentbackend`) CR.
- Il `TridentBackendConfig` è legato in modo univoco a un `TridentBackend` che è stato creato da Trident.

Ogni `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'effettivo oggetto backend.

ATTENZIONE	`TridentBackend` I CR sono creati automaticamente da Trident. Non dovresti modificarli. Se vuoi apportare aggiornamenti ai backend, fallo modificando l'oggetto `TridentBackendConfig`.

Vedere il seguente esempio per il formato del `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 anche possibile dare un'occhiata agli esempi nella directory "trident-installer" per configurazioni di esempio per la piattaforma di storage/servizio desiderato.

Il `spec` accetta parametri di configurazione specifici del backend. In questo esempio, il backend utilizza il driver di storage `ontap-san` e utilizza i parametri di configurazione qui tabulati. Per l'elenco delle opzioni di configurazione per il driver di storage desiderato, fare riferimento al "informazioni sulla configurazione del backend per il tuo storage driver".

La `spec` sezione include anche `credentials` e `deletionPolicy` campi, che sono stati introdotti di recente nella `TridentBackendConfig` CR:

- `credentials`: Questo parametro è un campo obbligatorio e contiene le credenziali utilizzate per l'autenticazione con il sistema storage/servizio. Questo è impostato su un Kubernetes Secret creato dall'utente. Le credenziali non possono essere passate in testo normale e genereranno un errore.
- `deletionPolicy`: Questo campo definisce cosa dovrebbe accadere quando `TridentBackendConfig` viene eliminato. Può assumere uno dei due valori possibili:
  - `delete`: Ciò comporta l'eliminazione sia di `TridentBackendConfig` CR che del backend associato. Questo è il valore predefinito.
  - `retain`: Quando un `TridentBackendConfig` CR viene eliminato, la definizione del backend sarà ancora presente e potrà essere gestita con `tridentctl`. Impostando la policy di eliminazione su `retain`, gli utenti potranno effettuare il downgrade a una versione precedente (pre-21.04) e mantenere i backend creati. Il valore di questo campo può essere aggiornato dopo che un `TridentBackendConfig` è stato creato.

NOTA	Il nome di un backend viene impostato usando `spec.backendName`. Se non specificato, il nome del backend viene impostato sul nome dell' `TridentBackendConfig`oggetto (`metadata.name`). Si consiglia di impostare esplicitamente i nomi dei backend usando `spec.backendName`.
SUGGERIMENTO	I backend creati con `tridentctl` non hanno un oggetto `TridentBackendConfig` associato. Puoi scegliere di gestire tali backend con `kubectl` creando un `TridentBackendConfig` CR. È necessario prestare attenzione a specificare parametri di configurazione identici (come `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName` e così via). Trident assocerà automaticamente il nuovo `TridentBackendConfig` creato con il backend preesistente.

Panoramica dei passaggi

Per creare un nuovo backend utilizzando `kubectl`, dovresti fare quanto segue:

1. Crea un "Kubernetes Secret". Il secret contiene le credenziali di cui Trident ha bisogno per comunicare con il cluster/servizio di storage.
2. Crea un `TridentBackendConfig` oggetto. Questo contiene informazioni specifiche sul cluster/servizio di storage e fa riferimento al segreto creato nel passaggio precedente.

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

Passaggio 1: crea un Kubernetes Secret

Crea un Secret che contiene le credenziali di accesso per il backend. Questo è univoco per ogni servizio/piattaforma 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 ogni storage platform:

Descrizione dei campi segreti della storage platform	Segreto	Descrizione dei campi
Azure NetApp Files	clientID	L'ID client da una registrazione dell'app
Element (NetApp HCI/SolidFire)	Punto finale	MVIP per il SolidFire cluster con credenziali tenant
ONTAP	nome utente	Nome utente per connettersi al cluster/SVM. Utilizzata per l'autenticazione basata sulle credenziali
ONTAP	password	Password per connettersi al cluster/SVM. Utilizzata per l'autenticazione basata sulle credenziali
ONTAP	clientPrivateKey	Valore codificato in Base64 della chiave privata del client. Utilizzato per l'autenticazione basata su certificato
ONTAP	chapUsername	Nome utente in entrata. Obbligatorio se useCHAP=true. Per `ontap-san` e `ontap-san-economy`
ONTAP	chapInitiatorSecret	Segreto dell'iniziatore CHAP. Obbligatorio se useCHAP=true. Per `ontap-san` e `ontap-san-economy`
ONTAP	chapTargetUsername	Nome utente di destinazione. Obbligatorio se useCHAP=true. Per `ontap-san` e `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	Segreto dell'iniziatore di destinazione CHAP. Obbligatorio se useCHAP=true. Per `ontap-san` e `ontap-san-economy`

Il Secret creato in questo passaggio verrà referenziato nel campo `spec.credentials` dell'oggetto `TridentBackendConfig` che viene creato nel passaggio successivo.

Passaggio 2: crea la `TridentBackendConfig` CR

Ora sei pronto per creare il tuo `TridentBackendConfig` CR. In questo esempio, un backend che utilizza il `ontap-san` driver viene creato utilizzando l'oggetto `TridentBackendConfig` 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 del `TridentBackendConfig` CR

Ora che hai creato la `TridentBackendConfig` CR, puoi verificarne 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 correttamente e associato al `TridentBackendConfig` CR.

La fase può assumere uno dei seguenti valori:

- `Bound`: Il `TridentBackendConfig` CR è associato a un backend e tale backend contiene `configRef` impostato sull' `TridentBackendConfig` uid del CR.
- `Unbound`: Rappresentato usando `""`. L' `TridentBackendConfig`oggetto non è vincolato a un backend. Tutte le nuove `TridentBackendConfig` CR sono in questa fase per impostazione predefinita. Dopo il cambio di fase, non può tornare a Unbound.
- `Deleting`: Il `TridentBackendConfig` CR `deletionPolicy` è stato impostato per l'eliminazione. Quando il `TridentBackendConfig` CR viene eliminato, passa allo stato Eliminazione in corso.
  - Se non esistono persistent volume claims (PVC) sul backend, l'eliminazione del `TridentBackendConfig` comporterà che Trident elimini sia il backend sia il

`TridentBackendConfig` CR.

  - Se uno o più PVC sono presenti sul backend, questo entra in uno stato di eliminazione. Il `TridentBackendConfig` CR successivamente entra anch'esso in fase di eliminazione. Il backend e `TridentBackendConfig` vengono eliminati solo dopo che tutti i PVC sono stati eliminati.

- `Lost`: Il backend associato al `TridentBackendConfig` CR è stato eliminato accidentalmente o deliberatamente e il `TridentBackendConfig` CR contiene ancora un riferimento al backend eliminato. Il `TridentBackendConfig` CR può comunque essere eliminato indipendentemente dal `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 il `tridentbackends.trident.netapp.io` CRD è mancante. Ciò potrebbe richiedere un intervento.

A questo punto, un backend è stato creato con successo! Ci sono diverse operazioni che possono essere gestite in aggiunta, come "aggiornamenti del backend ed eliminazioni del backend".

(Facoltativo) Passaggio 4: Ottieni maggiori dettagli

Puoi eseguire il seguente comando per ottenere maggiori informazioni sul tuo 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 anche 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 che è stato 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, l'utente ha modificato qualcosa in `spec`) o attivata da Trident (ad esempio, durante i riavvii di Trident). Può essere Success o Failed. `phase` rappresenta lo stato della relazione tra il `TridentBackendConfig` CR e il backend. Nell'esempio sopra, `phase` ha il valore Bound, il che significa che il `TridentBackendConfig` CR è associato al backend.

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

ATTENZIONE	Non è possibile aggiornare o eliminare un backend che contiene un oggetto associato `TridentBackendConfig` utilizzando `tridentctl`. Per comprendere i passaggi necessari per passare tra `tridentctl` e `TridentBackendConfig`, "vedi qui".

Gestisci i backend

Esegui la gestione del backend con kubectl

Scopri come eseguire operazioni di gestione del backend utilizzando `kubectl`.

Elimina un backend

Eliminando un `TridentBackendConfig`, si indica a Trident di eliminare/mantenere i backend (in base a `deletionPolicy`). Per eliminare un backend, assicurarsi che `deletionPolicy` sia impostato su delete. Per eliminare solo `TridentBackendConfig`, assicurarsi che `deletionPolicy` sia impostato su retain. Questo garantisce che il backend sia ancora presente e possa essere gestito utilizzando `tridentctl`.

Eseguire il seguente comando:

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

Trident non elimina i Kubernetes Secrets che erano in uso da `TridentBackendConfig`. L'utente Kubernetes è responsabile della pulizia dei secrets. È necessario prestare attenzione quando si eliminano i secrets. Dovresti eliminare i secrets solo se non sono in uso dai backends.

Visualizza i backend esistenti

Eseguire il seguente comando:

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

È anche possibile eseguire `tridentctl get backend -n trident` o `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`.

Aggiorna un backend

Possono esserci molteplici motivi per aggiornare un backend:

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

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

- È necessario aggiornare i parametri (ad esempio il nome dell'ONTAP SVM utilizzato).
  - È possibile aggiornare `TridentBackendConfig` gli oggetti direttamente tramite Kubernetes utilizzando il seguente comando:

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

- In alternativa, puoi apportare modifiche al CR `TridentBackendConfig` esistente utilizzando il seguente comando:

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

NOTA

- Se un aggiornamento del backend fallisce, il backend continua a mantenere l'ultima configurazione nota. Puoi visualizzare i log per determinarne la causa eseguendo `kubectl get tbc <tbc-name> -o yaml -n trident` o `kubectl describe tbc <tbc-name> -n trident`.
- Dopo aver identificato e corretto il problema con il file di configurazione, puoi rieseguire il comando di aggiornamento.

Esegui la gestione del backend con tridentctl

Scopri come eseguire operazioni di gestione del backend utilizzando `tridentctl`.

Crea un backend

Dopo aver creato un "file di configurazione backend", esegui il seguente comando:

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

Se la creazione del backend fallisce, si è verificato un errore nella configurazione del backend. Puoi visualizzare i log per determinarne la causa eseguendo il seguente comando:

```
tridentctl logs -n trident
```

Dopo aver identificato e corretto il problema con il file di configurazione, puoi semplicemente eseguire il `create` comando nuovamente.

Elimina un backend

Per eliminare un backend da Trident, procedere come segue:

1. Recupera il nome del backend:

```
tridentctl get backend -n trident
```

2. Elimina il backend:

```
tridentctl delete backend <backend-name> -n trident
```

NOTA	Se Trident ha eseguito il provisioning di volumi e snapshot da questo backend che sono ancora esistenti, l'eliminazione del backend impedisce il provisioning di nuovi volumi da parte sua. Il backend continuerà a esistere in uno stato di "Eliminazione".

Visualizza 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
```

Aggiorna un backend

Dopo aver creato un nuovo file di configurazione backend, eseguire il seguente comando:

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

Se l'aggiornamento del backend fallisce, si è verificato un errore nella configurazione del backend o hai tentato un aggiornamento non valido. Puoi visualizzare i log per determinarne la causa eseguendo il seguente comando:

```
tridentctl logs -n trident
```

Dopo aver identificato e corretto il problema con il file di configurazione, puoi semplicemente eseguire il `update` comando nuovamente.

Identificare le classi di storage che utilizzano un backend

Questo è un esempio del tipo di domande a cui è possibile rispondere con il JSON che `tridentctl` restituisce per gli oggetti backend. Questo utilizza l' `jq`utility, che è necessario installare.

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

Ciò vale anche per i backend creati utilizzando `TridentBackendConfig`.

Spostarsi tra le opzioni di gestione del backend

Scopri i diversi modi per gestire i backend in Trident.

Opzioni per la gestione dei backend

Con l'introduzione di `TridentBackendConfig`, gli amministratori hanno ora due modi unici per gestire i backend. Questo pone le seguenti domande:

- I backend creati utilizzando `tridentctl` possono essere gestiti con `TridentBackendConfig`?
- I backend creati utilizzando `TridentBackendConfig`possono essere gestiti utilizzando `tridentctl`?

Gestisci `tridentctl` backend utilizzando `TridentBackendConfig`

Questa sezione illustra i passaggi necessari per gestire i backend che sono stati creati utilizzando `tridentctl` direttamente tramite l'interfaccia Kubernetes creando `TridentBackendConfig` oggetti.

Questo si applicherà ai seguenti scenari:

- Backend preesistenti, che non hanno un `TridentBackendConfig` perché sono stati creati con `tridentctl`.
- Nuovi backend che sono stati creati con `tridentctl`, mentre esistono altri oggetti `TridentBackendConfig`.

In entrambi gli scenari, i backend continueranno a essere presenti, con Trident che pianifica i volumi e opera su di essi. Gli amministratori hanno due possibilità:

- Continua a utilizzare `tridentctl` per gestire i backend creati utilizzandolo.
- Associa i backend creati usando `tridentctl` a un nuovo `TridentBackendConfig` oggetto. Così facendo, i backend saranno gestiti usando `kubectl` e non `tridentctl`.

Per gestire un backend preesistente utilizzando `kubectl`, è necessario creare un `TridentBackendConfig` che si colleghi al backend esistente. Ecco una panoramica di come funziona:

1. Crea un segreto Kubernetes. Il segreto contiene le credenziali Trident necessarie per comunicare con il cluster/servizio di storage.
2. Crea un `TridentBackendConfig` oggetto. Questo contiene informazioni specifiche sul cluster/servizio di storage e fa riferimento al segreto creato nel passaggio precedente. È necessario prestare attenzione a 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.

Passaggio 0: Identificare il backend

Per creare un `TridentBackendConfig` che si leghi a un backend esistente, è necessario ottenere la configurazione del backend. In questo esempio, supponiamo che un backend sia stato creato 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"
      }
    }
  ]
}
```

Passaggio 1: crea un Kubernetes Secret

Crea un Secret che contiene le credenziali per il backend, come mostrato 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
```

Passaggio 2: crea un `TridentBackendConfig` CR

Il passaggio successivo consiste nel creare una `TridentBackendConfig` CR che si associ automaticamente a quella preesistente `ontap-nas-backend` (come in questo esempio). Assicurarsi che siano soddisfatti i seguenti requisiti:

- Lo stesso nome del backend è definito in `spec.backendName`.
- I parametri di configurazione sono identici al backend originale.
- I pool virtuali (se presenti) devono mantenere lo stesso ordine come nel backend originale.
- Le credenziali vengono fornite tramite un Kubernetes Secret e non in testo normale.

In questo caso, il `TridentBackendConfig` apparirà così:

```
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 del `TridentBackendConfig` CR

Dopo che la `TridentBackendConfig` è stata creata, la sua fase deve essere `Bound`. Dovrebbe inoltre riflettere lo stesso nome backend 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 backend sarà ora completamente gestito utilizzando l' `tbc-ontap-nas-backend` `TridentBackendConfig`oggetto.

Gestisci `TridentBackendConfig` backend utilizzando `tridentctl`

```
`tridentctl` può essere utilizzato per elencare i backend che sono stati
creati usando  `TridentBackendConfig`. Inoltre, gli amministratori possono
anche scegliere di gestire completamente tali backend tramite
`tridentctl` eliminando  `TridentBackendConfig` e assicurandosi che
`spec.deletionPolicy` sia impostato su  `retain`.
```

Passaggio 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].

Passaggio 1: confermare `deletionPolicy` **è impostato su** `retain`

Diamo un'occhiata al valore di `deletionPolicy`. Questo deve essere impostato su `retain`. Questo garantisce che quando un `TridentBackendConfig` CR viene eliminato, la definizione del 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 procedere al passaggio successivo a meno che `deletionPolicy` non sia impostato su `retain`.

Passaggio 2: Eliminare il `TridentBackendConfig` CR

Il passaggio finale consiste nell'eliminare il `TridentBackendConfig` CR. Dopo aver verificato che `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 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

Dopo l'eliminazione dell'oggetto `TridentBackendConfig`, Trident lo rimuove semplicemente senza eliminare effettivamente il backend stesso.

Crea e gestisci classi di archiviazione

Creare una storage class

Configura un oggetto Kubernetes StorageClass e crea la storage class per istruire Trident su come effettuare il provisioning dei volumi.

Configura un oggetto Kubernetes StorageClass

Il "Oggetto Kubernetes StorageClass" identifica Trident come il provisioner utilizzato per quella classe e istruisce Trident su come eseguire 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
```

Fate riferimento a "Oggetti Kubernetes e Trident" per i dettagli su come le classi di storage interagiscono con `PersistentVolumeClaim` e sui parametri per controllare come Trident effettua il provisioning dei volumi.

Creare una storage class

Dopo aver creato l'oggetto StorageClass, puoi creare la storage class. Esempi di storage class fornisce alcuni esempi di base che puoi utilizzare o modificare.

Passaggi

1. Si tratta di un oggetto Kubernetes, quindi usa `kubectl` per crearlo in Kubernetes.

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

2. Ora dovresti vedere una classe di storage **basic-csi** sia in Kubernetes che in Trident, e Trident dovrebbe aver rilevato i pool sul backend.

   ```
   kubectl get sc basic-csi
   ```

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

```
./tridentctl -n trident get storageclass basic-csi -o json
```

```
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

Esempi di storage class

Trident fornisce "definizioni di storage class semplici per specifici backend".

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

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.

Visualizza 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 storage class Kubernetes, eseguire il seguente comando:

```
kubectl get storageclass <storage-class> -o json
```

- Per visualizzare le classi di storage sincronizzate di Trident, eseguire il seguente comando:

```
tridentctl get storageclass
```

- Per visualizzare i dettagli della classe di archiviazione sincronizzata di Trident, eseguire il seguente comando:

```
tridentctl get storageclass <storage-class> -o json
```

Imposta una storage class predefinita

Kubernetes 1.6 ha aggiunto la possibilità di impostare una classe di storage predefinita. Questa è la classe di storage che verrà utilizzata per il provisioning di un Persistent Volume se un utente non ne specifica una in una Persistent Volume Claim (PVC).

- Definire una classe di archiviazione predefinita impostando l'annotazione `storageclass.kubernetes.io/is-default-class` a true nella definizione della classe di archiviazione. Secondo la specifica, qualsiasi altro valore o l'assenza dell'annotazione è interpretato come false.
- 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, puoi rimuovere l'annotazione della storage class predefinita utilizzando il seguente comando:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Ci sono anche esempi nel bundle di installazione di Trident che includono questa annotazione.

NOTA	Dovrebbe esserci solo una classe di storage predefinita nel tuo cluster alla volta. Kubernetes non ti impedisce tecnicamente di averne più di una, ma si comporterà come se non ci fosse affatto una classe di storage predefinita.

Identificare il backend per una storage class

Questo è un esempio del tipo di domande a cui è possibile rispondere con il JSON che `tridentctl` restituisce per gli oggetti backend di Trident. Questo utilizza l' `jq`utility, che potrebbe essere necessario installare prima.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

Elimina una storage class

Per eliminare una storage class da Kubernetes, esegui il seguente comando:

```
kubectl delete storageclass <storage-class>
```

`<storage-class>` dovrebbe essere sostituito con la tua storage class.

Tutti i volumi persistenti che sono stati creati tramite questa classe di archiviazione rimarranno intatti e Trident

continuerà a gestirli.

NOTA	Trident applica un valore vuoto `fsType` per i volumi che crea. Per i backend iSCSI, si consiglia di applicare `parameters.fsType` nella StorageClass. È necessario eliminare le StorageClasses esistenti e ricrearle con `parameters.fsType` specificato.

Effettua il provisioning e gestisci i volumi

Effettua il provisioning di un volume

Crea un PersistentVolumeClaim (PVC) che utilizza il StorageClass Kubernetes configurato per richiedere l'accesso al PV. Puoi quindi montare il PV su un pod.

Panoramica

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 PVC puoi montare il volume in un pod.

Crea il PVC

Passaggi

1. Crea 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. Monta il volume in un pod.

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

NOTA	Puoi monitorare l'avanzamento usando `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. Ora puoi eliminare il Pod. L'applicazione Pod non esisterà più, ma il volume rimarrà.

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

Esempi di manifest

PersistentVolumeClaim manifesti di esempio

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 StorageClass denominato `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 un StorageClass denominato `protection-gold`.

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

Esempi di manifest di pod

Questi esempi mostrano 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 di base NVMe/TCP

```
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
```

Fate riferimento a "Oggetti Kubernetes e Trident" per i dettagli su come le classi di storage interagiscono con `PersistentVolumeClaim` e sui parametri per controllare come Trident effettua il provisioning dei volumi.

Espandi volumi

Trident offre agli utenti Kubernetes la possibilità di espandere i propri volumi dopo che sono stati 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 provisioner CSI.

NOTA	L'espansione del volume iSCSI è supportata dai `ontap-san`, `ontap-san-economy`, `solidfire-san` driver e richiede Kubernetes 1.16 e versioni successive.

Passaggio 1: configurare la StorageClass per supportare l'espansione del volume

Modifica la definizione di StorageClass per impostare il campo `allowVolumeExpansion` su `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 il `allowVolumeExpansion` parametro.

Passaggio 2: crea un PVC con la StorageClass che hai creato

Modifica la definizione del PVC e aggiorna il `spec.resources.requests.storage` per riflettere la nuova dimensione desiderata, che deve essere maggiore della dimensione originale.

```
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 Persistent Volume (PV) e lo associa a questo Persistent Volume Claim (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 collega il PVC

Collega il PV a un pod per ridimensionarlo. Esistono due scenari quando si ridimensiona un PV iSCSI:

- Se il PV è collegato a un pod, Trident espande il volume sul backend di archiviazione, esegue una nuova scansione del dispositivo e ridimensiona il filesystem.
- Quando si tenta di ridimensionare un PV non collegato, Trident espande il volume sul backend di storage. Dopo che il PVC è stato associato a un pod, Trident esegue una nuova scansione del dispositivo e ridimensiona il filesystem. Kubernetes aggiorna la dimensione del PVC dopo che l'operazione di espansione è stata completata con successo.

In questo esempio, viene creato un pod che utilizza il `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 1Gi a 2Gi, modificare la definizione del PVC e aggiornare la `spec.resources.requests.storage` a 2Gi.

```
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: convalida l'espansione

È possibile verificare che l'espansione abbia funzionato correttamente controllando le dimensioni del PVC, del PV e del Trident volume:

```
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 FC Persistent Volume (PV) utilizzando il CSI provisioner.

NOTA L'espansione del volume FC è supportata dal driver `ontap-san` e richiede Kubernetes 1.16 e versioni successive.

Passaggio 1: configurare la StorageClass per supportare l'espansione del volume

Modifica la definizione di StorageClass per impostare il campo `allowVolumeExpansion` su `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 il `allowVolumeExpansion` parametro.

Passaggio 2: crea un PVC con la StorageClass che hai creato

Modifica la definizione del PVC e aggiorna il `spec.resources.requests.storage` per riflettere la nuova dimensione desiderata, che deve essere maggiore della dimensione originale.

```
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 Persistent Volume (PV) e lo associa a questo Persistent Volume Claim (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 collega il PVC

Collega il PV a un pod per ridimensionarlo. Esistono due scenari per il ridimensionamento di un PV FC:

- Se il PV è collegato a un pod, Trident espande il volume sul backend di archiviazione, esegue una nuova scansione del dispositivo e ridimensiona il filesystem.
- Quando si tenta di ridimensionare un PV non collegato, Trident espande il volume sul backend di storage. Dopo che il PVC è stato associato a un pod, Trident esegue una nuova scansione del dispositivo e ridimensiona il filesystem. Kubernetes aggiorna la dimensione del PVC dopo che l'operazione di espansione è stata completata con successo.

In questo esempio, viene creato un pod che utilizza il `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 1Gi a 2Gi, modificare la definizione del PVC e aggiornare la `spec.resources.requests.storage` a 2Gi.

```
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: convalida l'espansione

È possibile verificare che l'espansione abbia funzionato correttamente controllando le dimensioni del PVC, del PV e del Trident volume:

```
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.

Passaggio 1: configurare la StorageClass per supportare l'espansione del volume

Per ridimensionare un NFS PV, l'amministratore deve prima configurare la storage class per consentire l'espansione del volume impostando il `allowVolumeExpansion` field su `true`:

```
cat storageclass-ontapnas.yaml
```

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

Se hai già creato una classe di archiviazione senza questa opzione, puoi semplicemente modificare la classe

di archiviazione esistente utilizzando `kubectl edit storageclass` per consentire l'espansione del volume.

Passaggio 2: crea un PVC con la StorageClass che hai creato

```
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, modifica il PVC e imposta `spec.resources.requests.storage` su 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: convalida l'espansione

È possibile verificare che il ridimensionamento abbia funzionato correttamente controllando le dimensioni del PVC, del 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 introduce la scalabilità del controller attraverso una migliore concorrenza tra più driver di storage. I clienti possono identificare quali driver Trident supportano la scalabilità

del controller al momento della disponibilità generale e quali driver sono disponibili come anteprima tecnica in Trident 26.02. Questo consente decisioni di implementazione informate e una gestione appropriata dei rischi per ambienti Kubernetes scalabili.

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, 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 la scalabilità del controller è abilitata, le operazioni su volumi e backend diversi procedono contemporaneamente. Ciò aumenta il throughput e riduce i tempi operativi end-to-end in ambienti con un numero elevato di operazioni PersistentVolumeClaim e VolumeSnapshot simultanee.

Supporto per la scalabilità del controller

Trident supporta la scalabilità del controller con diversi livelli di maturità a seconda del driver di storage.

Disponibilità generale

I seguenti driver supportano la scalabilità del controller alla disponibilità generale in Trident 26.02:

- `ontap-san`
- `ontap-nas`
- `google-cloud-netapp-volumes`

NOTA	I `google-cloud-netapp-volumes` e `google-cloud-netapp-volumes-san` driver sono diversi. Solo `google-cloud-netapp-volumes` è supportato. Non utilizzare `google-cloud-netapp-volumes-san` nelle configurazioni di backend o negli esempi.

Abilita la scalabilità del controller

La scalabilità del controller è controllata dall'opzione di configurazione `enableConcurrency`. Questa opzione deve essere abilitata esplicitamente durante l'installazione di Trident o aggiornando una distribuzione esistente.

Distribuzione dell'operatore Trident

Per abilitare la scalabilità del controller con l'operatore Trident, impostare `enableConcurrency` su `true` nella `TridentOrchestrator` risorsa personalizzata.

Nuova installazione

Crea o modifica il `TridentOrchestrator` CR con `enableConcurrency` impostato su `true`:

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

Applica il 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}}'
```

Verificare che l'impostazione sia stata applicata:

```
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 values:

```
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 esistente basata su `tridentctl`, disinstallarla e reinstallarla 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, verificare che il controller Trident sia in esecuzione con la concorrenza abilitata controllando i log del pod del controller:

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

Dovresti vedere una voce di log che indica che la concorrenza è abilitata.

Anteprima tecnica

I seguenti driver supportano la scalabilità del controller come anteprima tecnica in Trident 26.02:

- `nas-eco`
- `san-eco`

Per questi driver:

- La concorrenza del controller è disponibile per la valutazione e il testing

- Il comportamento potrebbe cambiare nelle versioni future
- L'uso in ambienti di produzione non è raccomandato

Comportamento della concorrenza

Quando la scalabilità del controller è abilitata:

- Trident sostituisce il singolo blocco globale con un blocco a grana fine per risorsa
- Le operazioni che modificano la stessa risorsa vengono serializzate per mantenere la coerenza dei dati
- Le operazioni che leggono solo da una risorsa possono procedere 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 archiviazione backend
- Se più backend condividono lo stesso LIF di gestione, condividono questo limite di 20 richieste

Limitazioni e considerazioni note

Le seguenti considerazioni si applicano alla scalabilità del controller:

- La concorrenza è gestita internamente dal Trident controller
- In questa release non sono presenti limiti di concorrenza configurabili dall'utente
- Il throughput complessivo dipende da:
  - Il driver di archiviazione in uso
  - Reattività del backend
  - Prestazioni del server API Kubernetes
- Un'elevata concorrenza può aumentare il carico sui sistemi di storage back-end

Avvertenze e limitazioni

Le seguenti limitazioni si applicano in Trident 26.02:

- Il comportamento di scalabilità del controller non è identico in tutti i driver
- I driver di anteprima tecnica potrebbero presentare:
  - Prestazioni incoerenti sotto carico elevato
  - Modifiche nel comportamento tra le release
- Il debug delle operazioni concorrenti può essere più complesso a causa dell'esecuzione parallela
- Le metriche e i registri potrebbero mostrare output di operazioni intercalate

Raccomandazioni

- Utilizza driver di disponibilità generale (GA) per ambienti di produzione che richiedono elevata scalabilità
- Valuta i driver di anteprima tecnica in ambienti non di produzione
- Monitora le prestazioni del backend e del controller quando si opera su larga scala
- Evita di dare per scontato l'ordinamento 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

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	Richiesto	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

Passaggi

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.
Fallito	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`.

Passaggi

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	Usi `standard-agp`.

Annotazione StorageClass	Annotazione PVC	Comportamento efficace
`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 crescita automatica

```
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
```

Gestisci le policy di Autogrow

Dopo aver creato le Autogrow Policies, puoi visualizzarle, aggiornarle ed eliminarle secondo necessità. Puoi anche monitorare quali volumi stanno utilizzando 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 interessano tutti i volumi che attualmente utilizzano la policy. Testare prima le modifiche in un ambiente non di produzione, se possibile.

Passaggi

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 nuovi parametri alla successiva valutazione della crescita.
- **Non è necessario riavviare il volume:** Le modifiche si applicano alla successiva 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 Autogrow utilizzano la protezione del finalizzatore per impedire l'eliminazione accidentale mentre i volumi le stanno utilizzando attivamente.

Passaggi

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`.

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 (distribuzioni brownfield):** l'autogrow potrebbe non funzionare per i volumi esistenti anche se viene applicata una valida Autogrow Policy. Ciò è dovuto a una migrazione in corso delle pubblicazioni dei volumi. Per confermare che la migrazione sia stata completata, controlla i log del controller Trident per `"Migration completed"` messaggi.

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"`. Questo sovrascrive qualsiasi criterio 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 vengono create ulteriori richieste di ridimensionamento per quel volume, anche se l'utilizzo continua ad aumentare oltre il `usedThreshold`.

Ad esempio, con questa policy, Trident smette di aumentare il volume una volta raggiunti 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 si aggiorna una policy, tutti i volumi che la utilizzano adottano i nuovi parametri alla successiva valutazione della crescita. Non è necessario riavviare alcun volume.

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 `Deleting` stato e attende che tutti i volumi vengano rimossi 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 che tutti i volumi sono stati rimossi, il finalizzatore viene rilasciato e il criterio viene eliminato.

L'autogrow funziona con tutti i backend ONTAP?

Autogrow supporta i protocolli NFS, iSCSI, FCP e NVMe. Tuttavia, i volumi raw block NVMe richiedono ONTAP 9.16.1 o versioni successive.

I volumi esistenti nelle distribuzioni brownfield potrebbero richiedere il completamento della migrazione della pubblicazione del volume prima che l'autogrow abbia effetto. Verificare lo stato della migrazione controllando 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 volumi

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 per:

- Containerizzare un'applicazione e riutilizzare il set di dati esistente
- Utilizza un clone di un set di dati per un'applicazione effimera
- Ricostruire un cluster Kubernetes guasto
- Migrare i 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 (read-write). I volumi di tipo DP (data protection) sono volumi di destinazione di SnapMirror. È necessario 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 e poi eseguire l'importazione.

ATTENZIONE	Questo è particolarmente importante per i volumi a blocchi, poiché Kubernetes non sarebbe a conoscenza della connessione precedente e potrebbe facilmente collegare un volume attivo a un pod. Questo può causare la corruzione dei dati.

- Sebbene `StorageClass` debba essere specificato su un PVC, Trident non utilizza questo parametro durante l'importazione. Le classi di archiviazione vengono utilizzate durante la creazione del volume per selezionare tra i pool disponibili in base alle caratteristiche di archiviazione. Poiché il volume esiste già, durante l'importazione non è richiesta la selezione del pool. Pertanto, l'importazione non fallirà anche se il volume esiste su un backend o un pool che non corrisponde alla classe di archiviazione specificata nel PVC.
- La dimensione del volume esistente viene determinata e impostata nel PVC. Dopo che il volume è stato importato dal driver di archiviazione, il PV viene creato con un ClaimRef al PVC.
  - La politica di recupero è inizialmente impostata su `retain` nel PV. Dopo che Kubernetes esegue correttamente il binding del PVC e del PV, la politica di recupero viene aggiornata per corrispondere alla politica di recupero della Storage Class.
  - Se il criterio di recupero della Storage Class è `delete`, il volume di archiviazione verrà eliminato quando il PV viene eliminato.
- Per impostazione predefinita, Trident gestisce il PVC e rinomina il volume FlexVol e la LUN 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 si utilizza 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 il PV viene eliminato e altre operazioni come il clone del volume e il ridimensionamento del volume vengono anch'esse ignorate.
  - `--no-rename`* - Se si usa 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` (compresi i sistemi ASA r2) e `ontap-san-economy` driver.

SUGGERIMENTO	Queste opzioni sono utili se si desidera utilizzare Kubernetes per i carichi di lavoro containerizzati, ma altrimenti 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 ha il duplice scopo di 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

Passaggi

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 è 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. Usa il comando `tridentctl import` per specificare il nome del backend Trident contenente il volume e il nome che identifica in modo univoco il volume sullo storage (ad esempio: ONTAP FlexVol, Element Volume). L'argomento `-f` è necessario per specificare il percorso del file PVC.

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

Utilizzo delle annotazioni PVC

Passaggi

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. L'impostazione predefinita 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

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

ONTAP NAS e ONTAP NAS FlexGroup

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

NOTA

- Trident non supporta l'importazione di volumi utilizzando il `ontap-nas-economy` driver.
- I `ontap-nas` e `ontap-nas-flexgroup` driver non consentono nomi di volumi duplicati.

Ogni volume creato con il `ontap-nas` driver è un volume FlexVol sul cluster ONTAP. L'importazione di volumi FlexVol con il `ontap-nas` driver funziona allo stesso modo. Un volume FlexVol già esistente su un cluster ONTAP può essere importato come un `ontap-nas` PVC. Allo stesso modo, i volumi FlexGroup 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

L'esempio seguente importa 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.

Il seguente esempio importa `unmanaged_volume` sul `ontap_nas` backend:

```
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>
```

ONTAP SAN

Trident supporta l'importazione di volumi utilizzando i `ontap-san` (iSCSI, NVMe/TCP e FC) e `ontap-san-economy` driver.

Trident può importare volumi ONTAP SAN FlexVol che contengono una singola LUN. Questo è coerente con il driver `ontap-san`, che crea un volume FlexVol per ogni PVC e una LUN all'interno del volume FlexVol. Trident importa il volume FlexVol e lo associa alla definizione del PVC. Trident può importare volumi `ontap-san-economy` che contengono più LUN.

I seguenti esempi mostrano come importare volumi gestiti e non gestiti:

Volume gestito

Per i volumi gestiti, Trident rinomina il FlexVol volume nel `pvc-<uuid>` formato e il LUN all'interno del FlexVol volume in `lun0`.

Il seguente esempio importa 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

Il seguente esempio importa `unmanaged_example_volume` sul `ontap_san` backend:

```
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 hai LUN mappate su igroups che condividono un IQN con un IQN di un nodo Kubernetes, come mostrato nell'esempio seguente, riceverai l'errore: `LUN already mapped to initiator(s) in this group`. Dovrai rimuovere l'initiator o annullare la mappatura della LUN per importare il volume.

```
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 tramite il `solidfire-san` driver.

NOTA Il driver Element supporta nomi di volume duplicati. Tuttavia, Trident restituisce un errore se ci sono nomi di volume duplicati. Come soluzione alternativa, clona il volume, fornisci un nome di volume univoco e importa il volume clonato.

Il seguente esempio importa un `element-managed` volume sul backend `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 di Azure NetApp Files, identifica il volume tramite il suo percorso volume. Il percorso volume è la parte del percorso di esportazione del volume dopo il `:/`. Ad esempio, se il percorso di montaggio è `10.0.0.2:/importvol1`, il percorso volume è `importvol1`.

Il seguente esempio importa un `azure-netapp-files` volume sul backend `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
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

Il seguente esempio importa un `google-cloud-netapp-volumes` volume quando sono presenti due volumi nella stessa regione:

```
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

Con Trident, puoi assegnare nomi ed etichette significativi ai volumi che crei. Questo ti aiuta a identificare e mappare facilmente i volumi alle rispettive risorse Kubernetes (PVC). Puoi anche definire modelli a livello di backend per creare nomi di volumi personalizzati ed etichette personalizzate; tutti i volumi che crei, importi o cloni aderiranno ai modelli.

Prima di iniziare

Supporto per nomi ed etichette di volume personalizzabili:

- Operazioni di creazione, importazione e clonazione di 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 on-premises.
- Le etichette personalizzate sono supportate solo per i `ontap-san`, `ontap-nas` e `ontap-nas-flexgroup` driver.
- I nomi dei volumi 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 fallisce. Tuttavia, se l'applicazione del modello fallisce, il volume verrà nominato secondo la convenzione di

naming esistente.

- Il prefisso di archiviazione non è applicabile quando un volume viene nominato utilizzando un name template dalla configurazione del backend. Qualsiasi valore di prefisso desiderato può essere aggiunto direttamente al template.

Esempi di configurazione del backend con name template ed etichette

I modelli di nome personalizzati possono essere definiti a livello di root e/o pool.

Esempio di livello radice

```
{
  "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 a livello di 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 template 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 importazione di volumi, le etichette vengono aggiornate solo se il volume esistente ha 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 a livello di root nella definizione del backend.
3. Trident non supporta l'uso di un operatore slice con il prefisso dello storage.
4. Se i modelli non producono nomi di volume univoci, Trident aggiungerà alcuni caratteri casuali per creare nomi di volume univoci.
5. Se il nome personalizzato per un volume NAS economy supera i 64 caratteri, Trident nominerà i volumi in base alla convenzione di naming esistente. Per tutti gli altri driver ONTAP, se il nome del volume supera il limite di nome, il processo di creazione del volume non riesce.

Condividere un volume NFS tra namespace

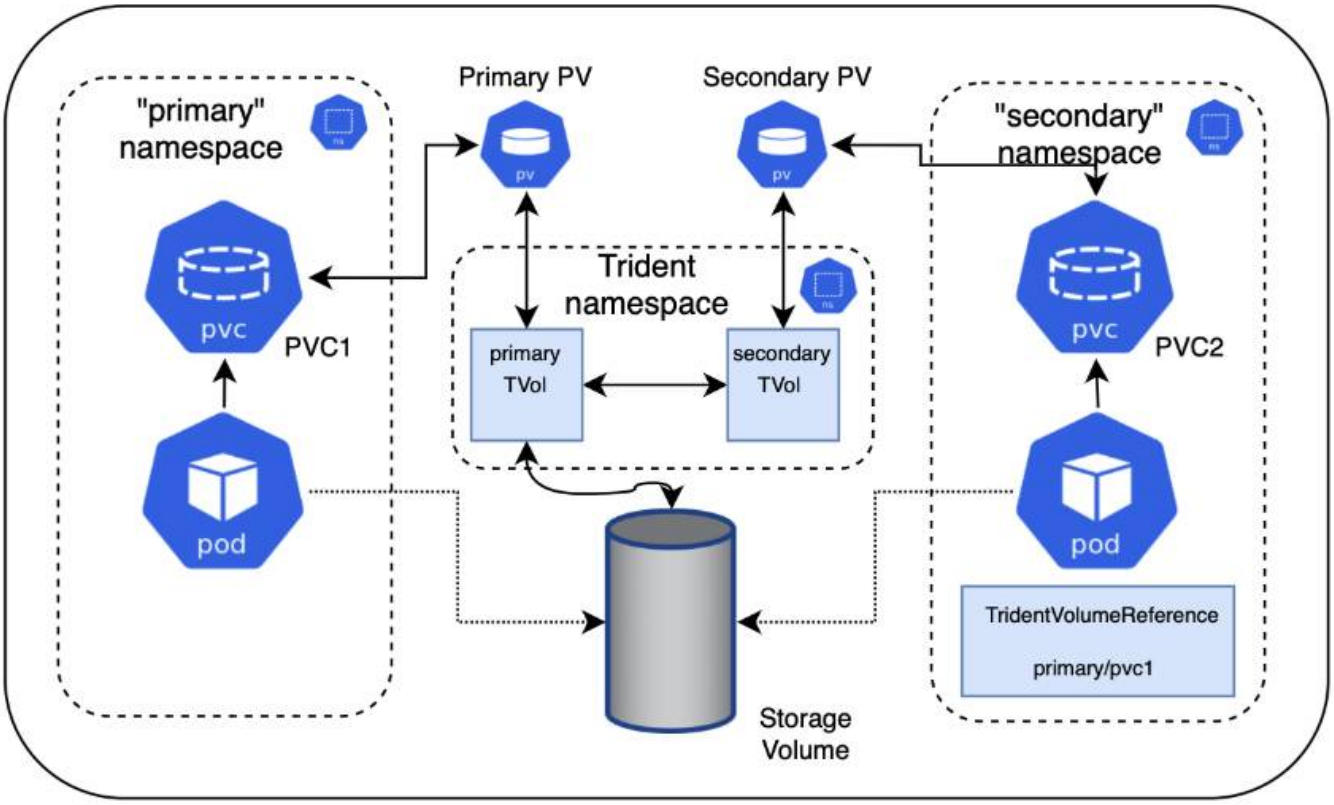
Utilizzando Trident, puoi creare un volume in uno spazio dei nomi primario e condividerlo in uno o più spazi dei nomi secondari.

Caratteristiche

La CR TridentVolumeReference consente di condividere in modo sicuro volumi NFS ReadWriteMany (RWX) su uno o più namespace Kubernetes. Questa soluzione nativa per Kubernetes offre i seguenti vantaggi:

- Più livelli di controllo degli accessi per garantire la sicurezza
- Funziona con tutti i driver di volume Trident NFS
- Nessuna dipendenza da tridentctl o da qualsiasi altra funzionalità non nativa di Kubernetes

Questo diagramma illustra la condivisione del volume NFS tra due namespace Kubernetes.



Avvio rapido

È possibile configurare la condivisione del volume NFS in pochi semplici passaggi.

1 **Configurare il PVC di origine per condividere il volume**

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

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

L'amministratore del cluster concede il permesso al proprietario dello spazio dei nomi di destinazione di creare il CR TridentVolumeReference.

3 **Creare TridentVolumeReference nello spazio dei nomi di destinazione**

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

4 **Crea 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 tra namespace richiede la collaborazione e l'azione del proprietario del namespace di origine, dell'amministratore del cluster e del proprietario del namespace di destinazione. Il ruolo dell'utente viene assegnato in ogni fase.

Passaggi

1. **Proprietario dello spazio dei nomi di origine:** crea il PVC (`pvc1`) nello spazio dei nomi di origine che concede l'autorizzazione alla condivisione con lo spazio dei nomi di destinazione (`namespace2`) utilizzando l'annotazione `shareToNamespace`.

```
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 archiviazione NFS backend.

   NOTA

   - È possibile condividere il PVC con più namespace utilizzando un elenco delimitato da virgole. Ad esempio, `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4`.
   - Puoi condividere con tutti gli spazi dei nomi utilizzando `*`. Ad esempio, `trident.netapp.io/shareToNamespace: *`
   - È possibile aggiornare il PVC per includere l `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 di creare il CR TridentVolumeReference nello spazio dei nomi di destinazione.
3. **Proprietario dello spazio dei nomi di destinazione:** Creare un TridentVolumeReference CR nello spazio dei nomi di destinazione che faccia riferimento 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 l'annotazione `shareFromPVC` per designare 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 a quella del PVC di origine.

Risultati

Trident legge l' `shareFromPVC`annotazione sul PVC di destinazione e crea il PV di destinazione come volume subordinato, senza risorse di storage proprie, che punta al PV di origine e condivide la risorsa di storage del PV di origine. Il PVC di destinazione e il PV di destinazione appaiono vincolati come di consueto.

Elimina un volume condiviso

È possibile eliminare un volume condiviso tra più namespace. Trident rimuoverà l'accesso al volume nel namespace di origine e manterrà l'accesso per gli altri namespace che condividono il volume. Quando tutti i namespace che fanno riferimento al volume vengono rimossi, Trident elimina il volume.

Utilizzare `tridentctl get` per interrogare i volumi subordinati

Utilizzando l'[`tridentctl`utility, puoi eseguire il `get`comando per ottenere i volumi subordinati. Per ulteriori informazioni, consultare `tridentctl` comandi e opzioni.

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

Flag:

- `-h, --help`: Aiuto per i volumi.
- `--parentOfSubordinate string`: Limita la query al volume sorgente subordinato.
- `--subordinateOf string`: Limita la query ai subordinati del volume.

Limitazioni

- Trident non può impedire ai namespace di destinazione di scrivere sul volume condiviso. È consigliabile utilizzare il blocco dei file o altri processi per impedire la sovrascrittura dei dati del volume condiviso.
- Non è possibile revocare l'accesso al PVC sorgente rimuovendo le `shareToNamespace` o `shareFromNamespace` annotazioni o eliminando il `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 saperne di più sull'accesso ai volumi tra namespace:

- Guarda la demo su "NetAppTV".

Clona volumi tra namespace

Utilizzando Trident, è possibile creare nuovi volumi utilizzando volumi esistenti o volumesnapshots da un namespace diverso all'interno dello stesso cluster Kubernetes.

Prerequisiti

Prima di clonare i volumi, assicurarsi 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 i driver di archiviazione `ontap-san` e `ontap-nas`. Le clonazioni di sola lettura non sono supportate.

Avvio rapido

È possibile configurare la clonazione dei volumi in pochi passaggi.

1. **Configura il PVC sorgente per clonare il volume**

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

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

L'amministratore del cluster concede il permesso al proprietario dello spazio dei nomi di destinazione di creare il CR TridentVolumeReference.

3 Creare TridentVolumeReference nello spazio dei nomi di destinazione

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

4 Creare il PVC clone nello spazio dei nomi di destinazione

Il proprietario dello spazio dei nomi di destinazione crea un 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, la clonazione di volumi tra spazi dei nomi 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 assegnato in ogni fase.

Passaggi

1. **Proprietario dello spazio dei nomi di origine:** Crea il PVC (`pvc1`) nello spazio dei nomi di origine (`namespace1`) che concede il permesso di condividere con lo spazio dei nomi di destinazione (`namespace2`) usando l'annotazione `cloneToNamespace`.

```
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 backend.

NOTA

- È possibile condividere il PVC con più namespace utilizzando un elenco delimitato da virgole. Ad esempio, `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4`.
- Puoi condividere con 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. **Cluster admin:** Assicurarsi che sia presente un RBAC appropriato per concedere il permesso al proprietario dello spazio dei nomi destinazione di creare il TridentVolumeReference CR nello spazio dei nomi destinazione (`namespace2`).
3. **Proprietario dello spazio dei nomi di destinazione:** Creare un TridentVolumeReference CR nello spazio dei nomi di destinazione che faccia riferimento 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:** Creare un PVC (`pvc2`) nello spazio dei nomi di destinazione (`namespace2`) usando le `cloneFromPVC` o `cloneFromSnapshot`, e `cloneFromNamespace` 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 provisionati utilizzando i driver ontap-nas-economy, i cloni di sola lettura non sono supportati.

Replicare i volumi utilizzando SnapMirror

Trident supporta le relazioni di mirroring tra un volume di origine su un cluster e il volume di destinazione sul cluster peered per replicare i dati per il disaster recovery. Puoi utilizzare una Custom Resource Definition (CRD) namespaced, chiamata Trident Mirror Relationship (TMR), per eseguire le seguenti operazioni:

- Crea relazioni di mirroring tra volumi (PVC)
- Rimuovere le relazioni di mirroring tra i volumi
- Interrompere le relazioni a specchio
- Promuovere il volume secondario durante le condizioni di disastro (failover)
- Esegui la transizione senza perdite delle applicazioni da un cluster a un altro cluster (durante i failover o le migrazioni pianificate)

Prerequisiti della replica

Assicurarsi che siano soddisfatti i seguenti prerequisiti prima di iniziare:

Cluster ONTAP

- **Trident**: Trident versione 22.10 o successiva deve essere presente sia sul cluster Kubernetes di origine che su quello di destinazione che utilizzano ONTAP come backend.
- **Licenze**: Le licenze asincrone ONTAP SnapMirror che utilizzano il bundle Data Protection devono essere abilitate sia sul cluster ONTAP di origine che su quello di destinazione. Consulta "Panoramica delle licenze SnapMirror in ONTAP" per ulteriori informazioni.

  A partire da ONTAP 9.10.1, tutte le licenze vengono fornite come NetApp license file (NLF), ovvero un singolo file che abilita più funzionalità. Consulta "Licenze incluse con ONTAP One" per maggiori informazioni.

NOTA	È supportata solo la protezione asincrona SnapMirror.

Peering

- **Cluster e SVM**: I backend di storage ONTAP devono essere sottoposti a peering. Consulta "Panoramica del peering di cluster e SVM" per ulteriori informazioni.

IMPORTANTE	Assicurarsi che i nomi SVM utilizzati nella relazione di replica tra due cluster ONTAP siano univoci.

- **Trident e SVM**: le SVM remote peered devono essere disponibili per Trident sul cluster di destinazione.

Driver supportati

NetApp Trident supporta la replicazione del volume con NetApp SnapMirror technology utilizzando classi di archiviazione supportate dai seguenti driver: **`ontap-nas`: NFS** `ontap-san`: iSCSI **`ontap-san`: FC** `ontap-san`: NVMe/TCP (richiede la versione di ONTAP minima 9.15.1)

NOTA	La replicazione del volume tramite SnapMirror non è supportata per i sistemi ASA r2. Per informazioni sui sistemi ASA r2, vedere "Scopri i sistemi di storage ASA r2".

Crea un PVC specchiato

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

Passaggi

1. Eseguire i seguenti passaggi sul cluster Kubernetes primario:
   a. Crea 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 la StorageClass creata in precedenza.

      Esempio

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

   c. Crea un MirrorRelationship CR 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 corrente di protezione dei dati (DP) del volume, quindi popola il campo di stato del MirrorRelationship.

d. Ottieni il TridentMirrorRelationship CR per ottenere il nome interno e l'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. Crea un 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. Crea un MirrorRelationship CR con informazioni sulla destinazione e sulla sorgente.

Esempio

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

Trident creerà una SnapMirror relazione con il nome della policy di relazione configurata (o predefinita per ONTAP) e la inizializzerà.

c. Creare un PVC con la StorageClass precedentemente creata per fungere da secondario (SnapMirror destinazione).

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 TridentMirrorRelationship CRD e non riuscirà a creare il volume se la relazione non esiste. Se la relazione esiste, Trident assicurerà che il nuovo FlexVol volume venga posizionato su una SVM in peering con la SVM remota definita nel MirrorRelationship.

Stati di replicazione del volume

Una Trident Mirror Relationship (TMR) è un CRD che rappresenta un'estremità di una relazione di replicazione tra PVC. La TMR di destinazione ha uno stato, che indica a Trident qual è lo stato desiderato. La TMR di destinazione ha i seguenti stati:

- **Stabilito**: il PVC locale è il volume di destinazione di una relazione mirror e questa è una nuova relazione.
- **Promosso**: il PVC locale è ReadWrite e montabile, con nessuna relazione speculare attualmente in vigore.
- **Ristabilito**: il PVC locale è il volume di destinazione di una relazione di SnapMirror ed era anche precedentemente in quella relazione di SnapMirror.
  - Lo stato ristabilito deve essere utilizzato se il volume di destinazione è mai stato in una relazione con il volume di origine, perché sovrascrive il contenuto del volume di destinazione.
  - Lo stato ripristinato non riuscirà se il volume non era precedentemente in una relazione con la sorgente.

Promuovere il PVC secondario durante un failover non pianificato

Eseguire il seguente passaggio sul cluster Kubernetes secondario:

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

Promuovere il PVC secondario durante un failover pianificato

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

Passaggi

1. Sul cluster Kubernetes primario, crea uno snapshot del PVC e attendi fino a quando lo snapshot viene creato.
2. Sul cluster Kubernetes primario, crea la CR SnapshotInfo per ottenere i dettagli interni.

   Esempio

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

3. Nel cluster Kubernetes secondario, aggiornare il campo *spec.state* del CR *TridentMirrorRelationship* su *promoted* e *spec.promotedSnapshotHandle* in modo che sia l'internalName dello snapshot.
4. Nel cluster Kubernetes secondario, confermare lo stato (campo status.state) di TridentMirrorRelationship su promoted.

Ripristina una relazione mirror dopo un failover

Prima di ripristinare una relazione speculare, scegli il lato che vuoi rendere come nuovo primario.

Passaggi

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

Operazioni aggiuntive

Trident supporta le seguenti operazioni sui volumi primario e secondario:

Replica il PVC primario in un nuovo PVC secondario

Assicurati di avere già un PVC primario e un PVC secondario.

Passaggi

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

Ridimensiona un PVC mirrorato, primario o secondario

Il PVC può essere ridimensionato normalmente, ONTAP espanderà automaticamente tutti i flevxols di destinazione se la quantità di dati supera la dimensione corrente.

Rimuovi la replicazione da un PVC

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

- Eliminare il MirrorRelationship sul PVC secondario. Questo interrompe la relazione di replicazione.
- Oppure, aggiorna il campo spec.state su *promoted*.

Elimina un PVC (che era stato precedentemente mirrorato)

Trident verifica la presenza di PVC replicati e rilascia la relazione di replicazione prima di tentare di eliminare il volume.

Elimina un TMR

L'eliminazione di un TMR su un lato di una relazione mirror fa sì che il TMR rimanente passi allo stato *promoted* prima che Trident completi l'eliminazione. Se il TMR selezionato per l'eliminazione è già nello stato *promoted*, non esiste alcuna relazione mirror e il TMR verrà rimosso e Trident promuoverà il PVC locale a *ReadWrite*. Questa eliminazione rilascia i metadati SnapMirror per il volume locale in ONTAP. Se questo volume verrà utilizzato in una relazione mirror in futuro, dovrà utilizzare un nuovo TMR con uno stato di replica del volume *established* durante la creazione della nuova relazione mirror.

Aggiorna le relazioni mirror quando ONTAP è online

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

Aggiorna le relazioni mirror quando ONTAP è offline

È possibile utilizzare un CRD per eseguire un aggiornamento SnapMirror senza che Trident abbia 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 TridentActionMirrorUpdate CRD. Può assumere un valore tra *Succeeded*, *In Progress* o *Failed*.

Usa la topologia CSI

Trident può creare e collegare selettivamente i volumi ai nodi presenti in un cluster Kubernetes facendo uso del "Funzione Topology CSI".

Panoramica

Utilizzando la funzionalità CSI Topology, l'accesso ai volumi può essere limitato a un sottoinsieme di nodi, in base alle regioni e alle zone di disponibilità. I provider di cloud oggi consentono agli amministratori di Kubernetes di creare nodi basati sulle zone. I nodi possono essere situati in diverse zone di disponibilità all'interno di una regione o in varie regioni. Per facilitare il provisioning dei volumi per i carichi di lavoro in un'architettura multizona, Trident utilizza CSI Topology.

SUGGERIMENTO	Scopri di più sulla funzione CSI Topology "qui".

Kubernetes offre due modalità uniche di binding dei volumi:

- Con `VolumeBindingMode` impostato su `Immediate`, Trident crea il volume senza alcuna consapevolezza della topologia. Il binding del volume e il provisioning dinamico vengono gestiti quando viene creato il PVC. Questo è il valore predefinito `VolumeBindingMode` ed è adatto per i cluster che non applicano vincoli topologici. I volumi persistenti vengono creati senza alcuna dipendenza dai requisiti di pianificazione del pod richiedente.
- Con `VolumeBindingMode` impostato su `WaitForFirstConsumer`, la creazione e il binding di un Persistent Volume per un PVC vengono ritardati 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 topologici.

NOTA	La `WaitForFirstConsumer` modalità di binding non richiede etichette di topologia. Questo può essere utilizzato indipendentemente dalla funzione Topologia CSI.

Cosa ti servirà

Per utilizzare la Topologia CSI, è necessario quanto segue:

- Un cluster Kubernetes che esegue un "versione supportata di Kubernetes"

```
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 del cluster dovrebbero avere etichette che introducono la consapevolezza della topologia (`topology.kubernetes.io/region` e `topology.kubernetes.io/zone`). Queste etichette **dovrebbero essere presenti sui nodi del cluster** prima che Trident sia installato affinché Trident sia consapevole della topologia.

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

Passo 1: Creare un backend consapevole della topologia

I backend di storage Trident possono essere progettati per fornire selettivamente volumi in base alle zone di disponibilità. Ogni backend può contenere un blocco opzionale `supportedTopologies` che rappresenta un elenco di zone e regioni supportate. Per le StorageClasses che fanno uso di un backend di questo tipo, un volume viene creato solo se richiesto da un'applicazione pianificata in una regione/zona supportata.

Ecco un esempio di definizione 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

```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` è usato per fornire un elenco di regioni e zone per backend. Queste regioni e zone rappresentano l'elenco dei valori consentiti che possono essere forniti in un StorageClass. Per StorageClasses che contengono un sottoinsieme delle regioni e zone fornite in un backend, Trident crea un volume sul backend.

È possibile definire `supportedTopologies` per pool di storage anche. 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, le etichette `region` e `zone` indicano la posizione del pool di storage. `topology.kubernetes.io/region` e `topology.kubernetes.io/zone` stabiliscono da dove possono essere consumati i pool di storage.

Passo 2: Definire StorageClasses che sono consapevoli della topologia

In base alle etichette topologiche fornite ai nodi nel cluster, StorageClasses può essere definito per contenere informazioni sulla topologia. Questo determinerà i pool di storage che servono come candidati per le richieste di PVC effettuate e il sottoinsieme di nodi che possono utilizzare i volumi forniti da Trident.

Vedere il seguente esempio:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions:
    - key: topology.kubernetes.io/zone
      values:
        - us-east1-a
        - us-east1-b
    - key: topology.kubernetes.io/region
      values:
        - us-east1
parameters:
  fsType: ext4
```

Nella definizione di StorageClass fornita sopra, `volumeBindingMode` è impostato su `WaitForFirstConsumer`. I PVC richiesti con questo StorageClass non saranno utilizzati finché non saranno referenziati in un pod. E, `allowedTopologies` fornisce le zone e la regione da utilizzare. Il `netapp-san-us-east1` StorageClass crea i PVC sul `san-backend-us-east1` backend definito sopra.

Fase 3: Creare e utilizzare un PVC

Con la StorageClass creata e mappata su un backend, ora puoi creare i PVC.

Vedi l'esempio `spec` qui 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 manifest avrebbe il seguente risultato:

```
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
```

Per consentire a Trident di creare un volume e associarlo al PVC, utilizzare 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 programmare il pod sui nodi presenti nella `us-east1` regione, e di scegliere tra qualsiasi nodo presente nella `us-east1-a` o `us-east1-b` zone.

Vedi 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 backend per includere `supportedTopologies`

I backend preesistenti possono essere aggiornati per includere un elenco di `supportedTopologies` utilizzando `tridentctl backend update`. Questo non influirà sui volumi che sono già stati forniti e sarà utilizzato solo per i PVC successivi.

Trova ulteriori informazioni

- "Gestisci le risorse per i contenitori"
- "nodeSelector"
- "Affinità e anti-affinità"
- "Taints e Tolleranze"

Lavora con gli snapshot

Gli snapshot dei volumi persistenti (PV) di Kubernetes consentono copie point-in-time dei volumi. È possibile creare uno snapshot di un volume creato utilizzando Trident, importare uno snapshot creato al di fuori di Trident, creare un nuovo volume da uno snapshot esistente e recuperare i dati del volume dagli snapshot.

Panoramica

Lo snapshot del volume è supportato dai `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files` e `google-cloud-netapp-volumes` driver.

Prima di iniziare

Per lavorare con gli snapshot, è necessario disporre di un controller snapshot esterno e di Custom Resource Definitions (CRD). Questa è responsabilità dell'orchestratore Kubernetes (ad esempio: Kubeadm, GKE, OpenShift).

Se la tua distribuzione Kubernetes non include il controller snapshot e i CRD, consulta Distribuire un controller snapshot del volume.

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

Crea uno Snapshot del volume

Passaggi

1. Crea un `VolumeSnapshotClass`. Per ulteriori informazioni, fare riferimento a "VolumeSnapshotClass".
   - Il `driver` punta al driver Trident CSI.
   - `deletionPolicy` può essere `Delete` o `Retain`. Quando impostato su `Retain`, lo snapshot fisico sottostante sullo storage cluster viene mantenuto anche quando l'oggetto `VolumeSnapshot` viene 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. Crea una Snapshot di un PVC esistente.

   Esempi

   - Questo esempio crea uno snapshot 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
     ```

   - In questo esempio viene creato un oggetto snapshot del volume per un PVC denominato `pvc1` e il nome dello snapshot è impostato su `pvc1-snap`. Un VolumeSnapshot è analogo a un PVC ed è associato a un oggetto `VolumeSnapshotContent` 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 l `VolumeSnapshotContent'oggetto per il `pvc1-snap` VolumeSnapshot descrivendolo. L `Snapshot Content Name'oggetto identifica l'oggetto VolumeSnapshotContent che serve questa snapshot. Il parametro `Ready To Use` indica che la snapshot 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
...
```

Crea un PVC da una Snapshot del volume

Puoi usare `dataSource` per creare un PVC usando un VolumeSnapshot denominato `<pvc-name>` come origine dei dati. Dopo che il PVC è stato creato, può essere collegato a un pod e usato come qualsiasi altro PVC.

ATTENZIONE	Il PVC verrà creato nello stesso backend del volume sorgente. Consulta "KB: Creazione di un PVC da un Trident PVC Snapshot non può essere creata in un backend alternativo".

L'esempio seguente crea il PVC utilizzando `pvc1-snap` come origine 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
```

Importa una Snapshot del volume

Trident supporta la "Processo di snapshot pre-provisionato Kubernetes" per consentire all'amministratore del cluster di creare un oggetto `VolumeSnapshotContent` e importare Snapshot creati al di fuori di Trident.

Prima di iniziare

Trident deve aver creato o importato il volume d'origine dello Snapshot.

Passaggi

1. **Cluster admin:** Creare un `VolumeSnapshotContent` oggetto che fa riferimento allo snapshot del backend. Questo avvia il flusso di lavoro dello snapshot in Trident.
   - Specificare il nome dello snapshot del backend in `annotations` come `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
   - Specificare `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` in `snapshotHandle`. Questa è l'unica informazione fornita a Trident dallo snapshotter esterno nella chiamata `ListSnapshots`.

     NOTA Il `<volumeSnapshotContentName>` non può sempre corrispondere al nome dello snapshot del backend a causa dei vincoli di denominazione del CR.

     Esempio

     Il seguente esempio crea un oggetto `VolumeSnapshotContent` che fa riferimento allo snapshot del 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:** Crea il `VolumeSnapshot` CR che fa riferimento all' `VolumeSnapshotContent` oggetto. Questa richiesta consente l'accesso all'utilizzo di `VolumeSnapshot` in un determinato namespace.

   Esempio

   Il seguente esempio crea un `VolumeSnapshot` CR chiamato `import-snap` che fa riferimento al `VolumeSnapshotContent` chiamato `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 snapshotter esterno riconosce il nuovo `VolumeSnapshotContent` e esegue la chiamata `ListSnapshots`. Trident crea il `TridentSnapshot`.
   - L'external snapshotter imposta `VolumeSnapshotContent` su `readyToUse` e `VolumeSnapshot` su `true`.
   - Trident restituisce `readyToUse=true`.
4. **Qualsiasi utente:** Crea un `PersistentVolumeClaim` per fare riferimento al nuovo `VolumeSnapshot`, dove il `spec.dataSource` (o `spec.dataSourceRef`) nome è il `VolumeSnapshot` nome.

   Esempio

Il seguente esempio crea un PVC che fa riferimento al `VolumeSnapshot` denominato `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
```

Recupera i dati del volume utilizzando le Snapshot

La directory delle snapshot è nascosta per impostazione predefinita per facilitare la massima compatibilità dei volumi forniti utilizzando i driver `ontap-nas` e `ontap-nas-economy`. Abilita la directory `.snapshot` per recuperare i dati direttamente dalle snapshot.

Utilizzare il comando ONTAP CLI di ripristino dell'istantanea del volume per ripristinare un volume a uno stato registrato in una precedente Snapshot.

```
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 vengono perse.

Ripristino del volume in loco da una Snapshot

Trident offre un rapido ripristino in-place del volume da una snapshot utilizzando il `TridentActionSnapshotRestore` (TASR) CR. Questo CR funziona come un'azione Kubernetes imperativa e non persiste dopo il completamento dell'operazione.

Trident supporta il ripristino delle snapshot sui driver `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes` e `solidfire-san`.

Prima di iniziare

È necessario disporre di un PVC vincolato e di una Snapshot del volume disponibile.

- Verificare che lo stato del PVC sia bound.

```
kubectl get pvc
```

- Verificare che la snapshot del volume sia pronta per l'uso.

```
kubectl get vs
```

Passaggi

1. Crea il CR TASR. Questo esempio crea un CR per PVC `pvc1` e volume snapshot `pvc1-snapshot`.

   NOTA Il TASR CR deve trovarsi in uno spazio dei nomi in cui esistono il PVC e il 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. Applica il CR per ripristinare dallo snapshot. Questo esempio ripristina dallo snapshot `pvc1`.

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

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

Risultati

Trident ripristina i dati dallo snapshot. Puoi 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 riproverà automaticamente l'operazione in caso di fallimento. Dovrai eseguire nuovamente l'operazione.
- Gli utenti Kubernetes senza accesso admin potrebbero dover ottenere il permesso dall'admin per creare un TASR CR nel loro namespace dell'applicazione.

Eliminare un PV con le relative Snapshot

Quando si elimina un Persistent Volume con le relative Snapshot, il volume Trident corrispondente viene aggiornato a uno "Deleting state". Rimuovere le Snapshot del volume per eliminare il volume Trident.

Distribuire un controller snapshot del volume

Se la tua distribuzione di Kubernetes non include il controller di snapshot e i CRD, puoi distribuirli come segue.

Passaggi

1. Crea CRD di Snapshot del 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. Crea 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, apri `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e aggiorna `namespace` al tuo namespace.

Link correlati

- "Istantanee del volume"
- "VolumeSnapshotClass"

Lavorare con le Snapshot dei gruppi di volumi

Snapshot del gruppo di volumi Kubernetes dei volumi persistenti (PV) NetApp Trident offre la possibilità di creare snapshot di più volumi (un gruppo di snapshot di volumi). Questo snapshot del gruppo di volumi rappresenta copie di più volumi acquisite nello stesso point-in-time.

NOTA VolumeGroupSnapshot is a beta feature in Kubernetes con API beta. Kubernetes 1.32 è la versione minima richiesta per VolumeGroupSnapshot.

Crea 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 superiore.
- Per lavorare con gli snapshot, è necessario disporre di un controller snapshot esterno e di Custom Resource Definitions (CRD). Questa è responsabilità dell'orchestratore Kubernetes (ad esempio: Kubeadm, GKE, OpenShift).

  Se la tua distribuzione Kubernetes non include il controller snapshot esterno e i CRD, consulta Distribuire un controller snapshot del volume.

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

- Nel file YAML del controller snapshot, impostare il `CSIVolumeGroupSnapshot` feature gate su 'true' per garantire che la funzionalità di snapshot del gruppo di volumi sia abilitata.
- Crea le classi di snapshot del gruppo di volumi richieste prima di creare uno snapshot del gruppo di volumi.
- Assicurati che tutti i PVC/volumi siano sullo stesso SVM per poter creare VolumeGroupSnapshot.

Passaggi

- Crea un VolumeGroupSnapshotClass prima di creare un VolumeGroupSnapshot. Per ulteriori informazioni, fai riferimento a "VolumeGroupSnapshotClass".

```
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
```

- Crea PVC con le etichette richieste utilizzando le classi di storage esistenti oppure aggiungi queste etichette ai PVC esistenti.

  L'esempio seguente crea il PVC utilizzando `pvc1-group-snap` come origine dati ed etichetta `consistentGroupSnapshot: groupA`. 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` specificata nel PVC.

  Questo esempio crea una Snapshot di un 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 una Snapshot di gruppo

È possibile ripristinare i singoli Persistent Volumes utilizzando le singole Snapshot che sono state create come parte del Volume Group Snapshot. Non è possibile recuperare il Volume Group Snapshot come unità.

Utilizzare il comando ONTAP CLI di ripristino dell'istantanea del volume per ripristinare un volume a uno stato registrato in una precedente Snapshot.

```
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 vengono perse.

Ripristino del volume in loco da una Snapshot

Trident offre un rapido ripristino in-place del volume da una snapshot utilizzando il `TridentActionSnapshotRestore` (TASR) CR. Questo CR funziona come un'azione Kubernetes imperativa e non persiste dopo il completamento dell'operazione.

Per ulteriori informazioni, vedi "Ripristino del volume in loco da una Snapshot".

Eliminare un PV con le Snapshot di gruppo associate

Quando si elimina una Snapshot di un volume di gruppo:

- È possibile eliminare VolumeGroupSnapshots come un tutto, non le singole snapshot nel gruppo.
- Se i PersistentVolumes vengono eliminati mentre esiste una snapshot per quel PersistentVolume, Trident sposterà quel volume in uno stato di "eliminazione" perché la snapshot deve essere rimossa prima che il volume possa essere rimosso in modo sicuro.
- Se è stato creato un clone utilizzando una snapshot raggruppata e poi il gruppo deve essere eliminato, inizierà un'operazione di split-on-clone e il gruppo non potrà essere eliminato fino al completamento della divisione.

Distribuire un controller snapshot del volume

Se la tua distribuzione di Kubernetes non include il controller di snapshot e i CRD, puoi distribuirli come segue.

Passaggi

1. Crea CRD di Snapshot del 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. Crea 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, apri `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e aggiorna `namespace` al tuo namespace.

Link correlati

- "VolumeGroupSnapshotClass"
- "Istantanee del volume"

Gestisci e monitora Trident

Aggiorna Trident

Aggiorna Trident

A partire dalla versione 24.02, Trident segue una cadenza di rilascio di quattro mesi, rilasciando tre release principali ogni anno solare. Ogni nuova release si basa sulle precedenti e offre nuove funzionalità, miglioramenti delle prestazioni, correzioni di bug e miglioramenti. Ti invitiamo ad aggiornare almeno una volta all'anno per sfruttare le nuove funzionalità di Trident.

Considerazioni prima dell'aggiornamento

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

- Dovrebbe essere installata una sola istanza di Trident in tutti gli spazi dei nomi in un dato cluster Kubernetes.
- Trident 23.07 e versioni successive richiedono snapshot del volume v1 e non supportano più snapshot alpha o beta.
- Durante l'aggiornamento, è importante fornire `parameter.fsType` in `StorageClasses` utilizzati da Trident. È possibile eliminare e ricreare `StorageClasses` senza interrompere i volumi preesistenti.
  - Questo è un **requisito** per l'applicazione "contesti di sicurezza" per i volumi SAN.
  - La directory sample input contiene esempi, come `storage-class-basic.yaml.templ` e `storage-class-bronze-default.yaml`.
  - Per ulteriori informazioni, fai riferimento a "Problemi noti".

Passaggio 1: seleziona una versione

Le versioni Trident 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 24.06 (o da qualsiasi 24.06 dot release) alla 25.06.
- 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 23.07 e desideri eseguire l'aggiornamento alla versione 25.06:
  a. Primo upgrade dal 23.07 al 24.06.
  b. Quindi eseguire l'aggiornamento da 24.06 a 25.06.

NOTA	Quando si esegue l'aggiornamento utilizzando l'operatore Trident su OpenShift Container Platform, è necessario eseguire l'aggiornamento a Trident 21.01.1 o versioni successive. L'operatore Trident rilasciato con 21.01.0 contiene un problema noto che è stato risolto in 21.01.1. Per ulteriori dettagli, consultare il "dettagli del problema su GitHub".

Fase 2: determinare il metodo di installazione originale

Per determinare quale versione hai utilizzato per installare originariamente Trident:

1. Usa `kubectl get pods -n trident` per esaminare i pod.
   - Se non è presente alcun pod operatore, Trident è stato installato usando `tridentctl`.
   - Se è presente un pod operatore, Trident è stato installato utilizzando l'operatore Trident manualmente o tramite Helm.
2. Se è presente un operator pod, usa `kubectl describe torc` per determinare se Trident è stato installato usando 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: Seleziona un metodo di aggiornamento

In genere, dovresti eseguire l'aggiornamento utilizzando lo stesso metodo che hai usato per l'installazione iniziale, tuttavia puoi "passare da un metodo di installazione all'altro". Ci sono due opzioni per aggiornare Trident.

- "Aggiorna utilizzando l'operatore Trident"

SUGGERIMENTO	Si consiglia di esaminare "Comprendere il workflow di upgrade dell'operatore" prima di eseguire l'aggiornamento con l'operatore.

*

Aggiorna con l'operatore

Comprendere il workflow di upgrade dell'operatore

Prima di utilizzare l'operatore Trident per aggiornare Trident, è necessario comprendere i processi in background che si verificano durante l'aggiornamento. Ciò include le modifiche al Trident controller, al controller Pod e ai node Pods, e al node DaemonSet che abilitano gli aggiornamenti continui.

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 "*aggiornamenti rolling*" offline. In particolare, l'operatore Trident comunica con il cluster Kubernetes per:

- Eliminare e ricreare la distribuzione del Trident Controller e il nodo DaemonSet.
- Sostituisci i Trident Controller Pod e i Trident Node Pod con nuove versioni.
  - Se un nodo non viene aggiornato, non impedisce che i nodi rimanenti vengano aggiornati.
  - Solo i nodi con un Trident Node Pod in esecuzione possono montare volumi.

SUGGERIMENTO	Per ulteriori informazioni sull'architettura Trident sul cluster Kubernetes, fare riferimento a "Architettura di 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 CRDs, RBAC e Trident SVC.
   c. Elimina la distribuzione del Trident Controller per la versione *n*.
   d. Crea la distribuzione del Trident Controller per la versione *n+1*.
2. **Kubernetes** crea il Trident Controller Pod per *n+1*.
3. L'operatore **Trident**:
   a. Elimina il DaemonSet del nodo Trident per *n*. L'operatore non attende la terminazione del Node Pod.
   b. Crea il Trident Node Daemonset per *n+1*.
4. **Kubernetes** crea Trident Node Pod sui nodi che non eseguono Trident Node Pod *n*. Questo garantisce che non ci sia mai più di un Trident Node Pod, di qualsiasi versione, su un nodo.

Aggiorna 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 "Seleziona un metodo di aggiornamento" prima di aggiornare un'installazione dell'operatore Trident.

Aggiorna un'installazione manuale

È possibile effettuare l'aggiornamento da un'installazione dell'operatore Trident con ambito cluster a un'altra installazione dell'operatore Trident con ambito cluster. Tutte le versioni di Trident utilizzano un operatore con ambito cluster.

NOTA	Per eseguire l'aggiornamento da Trident installato utilizzando l'operatore con ambito namespace (versioni 20.07 fino a 20.10), utilizzare le istruzioni di aggiornamento per "la tua versione installata" di Trident.

Informazioni su questa attività

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

- Per i cluster che eseguono Kubernetes 1.25 o versioni successive, usa "bundle_post_1_25.yaml".

Prima di iniziare

Assicurati di utilizzare un cluster Kubernetes che esegue "una versione supportata di Kubernetes".

Passaggi

1. Verifica la tua versione di Trident:

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

2. Aggiorna `operator.yaml`, `tridentorchestrator_cr.yaml` e `post_1_25_bundle.yaml` con il registry e i percorsi delle immagini per la versione a cui stai eseguendo l'aggiornamento (ad esempio 25.06) e il secret corretto.
3. Elimina l'operatore Trident che è stato utilizzato per installare l'istanza corrente di Trident. Ad esempio, se stai eseguendo l'aggiornamento dalla versione 25.02, esegui il seguente comando:

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. Se hai personalizzato l'installazione iniziale utilizzando `TridentOrchestrator` attributi, puoi modificare l'oggetto `TridentOrchestrator` per modificare i parametri di installazione. Questo potrebbe includere modifiche apportate per specificare registri di immagini Trident e CSI con mirroring per la modalità offline, abilitare i log 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 tua versione di Kubernetes. Ad esempio, se stai installando Trident 25.06.0, esegui il seguente comando:

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. Modifica il trident torc per includere l'immagine 25.06.0.

Aggiornare un'installazione di Helm

È possibile aggiornare un'installazione di Trident Helm.

ATTENZIONE	Quando si aggiorna un cluster Kubernetes dalla versione 1.24 alla 1.25 o successiva che ha Trident installato, è necessario aggiornare values.yaml per impostare `excludePodSecurityPolicy` su `true` o aggiungere `--set excludePodSecurityPolicy=true` al comando `helm upgrade` prima di poter aggiornare il cluster.

Se hai già aggiornato il tuo cluster Kubernetes dalla versione 1.24 alla 1.25 senza aggiornare il Trident helm, l'upgrade di helm fallisce. Perché l'upgrade di helm vada a buon fine, esegui questi passaggi come prerequisiti:

1. Installa il plugin helm-mapkubeapis da https://github.com/helm/helm-mapkubeapis.
2. Eseguire una simulazione per la release di Trident nello spazio dei nomi in cui Trident è installato. Questo elenca le risorse che verranno ripulite.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. Eseguire una corsa completa con helm per effettuare la pulizia.

```
helm mapkubeapis trident  --namespace trident
```

Passaggi

1. Se "installato Trident usando Helm", puoi usare `helm upgrade trident netapp-trident/trident-operator --version 100.2602.0` per aggiornare in un unico passaggio. Se non hai aggiunto il Helm repo o non puoi usarlo per aggiornare:
   a. Scarica l'ultima release di Trident da "la sezione *Assets* su GitHub".
   b. Usa il `helm upgrade` comando dove `trident-operator-26.02.0.tgz` riflette la versione a cui vuoi eseguire l'aggiornamento.

```
helm upgrade <name> trident-operator-26.02.0.tgz
```

> **NOTA** Se imposti opzioni personalizzate durante l'installazione iniziale (ad esempio specificando registri privati o mirror per le immagini di Trident e CSI), aggiungi il comando `helm upgrade` usando `--set` per assicurarti che tali opzioni siano incluse nel comando di aggiornamento, altrimenti i valori verranno reimpostati su quelli predefiniti.

2. Esegui `helm list` per verificare che la versione del chart e dell'app siano state entrambe aggiornate. Esegui `tridentctl logs` per rivedere eventuali messaggi di debug.

Aggiorna da un'installazione `tridentctl` a Trident operator

Puoi eseguire l'aggiornamento all'ultima release 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 "Spostarsi tra i metodi di installazione".

Passaggi

1. Scarica l'ultima release di Trident.

```
# Download the release required [26.02.0]
mkdir 26.02.0
cd 26.02.0
wget
https://github.com/NetApp/trident/releases/download/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

2. Crea il `tridentorchestrator` CRD dal manifesto.

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

3. Distribuisci 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. Crea una `TridentOrchestrator` CR per installare 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. Conferma che Trident è stato aggiornato alla versione prevista.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v26.02.0
```

Aggiorna con tridentctl

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

Informazioni su questa attività

La disinstallazione e la reinstallazione di Trident agiscono come un aggiornamento. Quando si disinstalla Trident, il Persistent Volume Claim (PVC) e il Persistent Volume (PV) utilizzati dalla distribuzione di Trident non vengono eliminati. I PV che sono già stati forniti rimarranno disponibili mentre Trident è offline e Trident fornirà volumi per qualsiasi PVC creato nel frattempo dopo che è tornato online.

Prima di iniziare

Rivedi "Seleziona un metodo di aggiornamento" prima di aggiornare usando `tridentctl`.

Passaggi

1. Esegui 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. Consulta "Installare Trident usando tridentctl".

IMPORTANTE	Non interrompere il processo di aggiornamento. Assicurati che il programma di installazione venga eseguito fino al completamento.

Gestisci Trident usando tridentctl

Il "Bundle di installazione Trident" include l'utilità a riga di comando `tridentctl` per fornire un accesso semplice a Trident. Gli utenti di Kubernetes con privilegi sufficienti possono usarla per installare Trident o gestire lo spazio dei nomi che contiene il pod Trident.

Comandi e flag globali

È possibile eseguire `tridentctl help` per ottenere un elenco dei 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`
Aggiungi una risorsa a Trident.

`delete`
Rimuovi una o più risorse da Trident.

`get`
Ottieni una o più risorse da Trident.

`help`
Guida su qualsiasi comando.

`images`
Stampa una tabella delle immagini dei container di cui Trident ha bisogno.

`import`
Importa una risorsa esistente in Trident.

`install`
Installa Trident.

`logs`
Stampa i log da Trident.

`send`
Invia una risorsa da Trident.

`uninstall`
Disinstalla Trident.

`update`
Modifica una risorsa in Trident.

`update backend state`
Sospendere temporaneamente le operazioni di backend.

`upgrade`
Aggiorna una risorsa in Trident.

`version`
Stampa la versione di Trident.

Bandiere globali

`-d, --debug`
Output di debug.

`-h, --help`
Aiuto per `tridentctl`.

`-k, --kubeconfig string`
Specificare il `KUBECONFIG` percorso per eseguire i comandi localmente o da un cluster Kubernetes a un altro.

NOTA	In alternativa, puoi esportare la `KUBECONFIG` variabile per puntare a uno specifico cluster Kubernetes e inviare `tridentctl` comandi a quel cluster.

`-n, --namespace string`
Namespace della distribuzione Trident.

`-o, --output string`
Formato di output. Uno tra json|yaml|name|wide|ps (predefinito).

`-s, --server string`
Indirizzo/porta dell'interfaccia REST di Trident.

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

Opzioni e flag dei comandi

crea

Utilizza il `create` comando per aggiungere una risorsa a Trident.

```
tridentctl create [option]
```

Opzioni

`backend`: Aggiungi un backend a Trident.

eliminare

Utilizzare il comando `delete` per rimuovere una o più risorse da Trident.

```
tridentctl delete [option]
```

Opzioni

`backend`: Elimina uno o più backend di storage da Trident.
`snapshot`: Elimina uno o più snapshot di volume da Trident.
`storageclass`: Elimina una o più storage class da Trident.

`volume`: Elimina uno o più storage volume da Trident.

get

Utilizza il `get` comando per ottenere una o più risorse da Trident.

```
tridentctl get [option]
```

Opzioni

`backend`: Ottieni uno o più backend di storage da Trident.
`snapshot`: Ottieni uno o più snapshot da Trident.
`storageclass`: Ottieni una o più storage class da Trident.
`volume`: Ottieni uno o più volumi da Trident.

Flag

`-h`, `--help`: Guida per i volumi.
`--parentOfSubordinate string`: Limita la query al volume sorgente subordinato.
`--subordinateOf string`: Limita la query ai subordinati del volume.

immagini

Utilizza `images` i flag per stampare una tabella delle immagini dei container di cui Trident ha bisogno.

```
tridentctl images [flags]
```

Flag

`-h`, `--help`: Aiuto per le immagini.
`-v`, `--k8s-version string`: Versione semantica del cluster Kubernetes.

importa volume

Utilizzare il comando `import volume` per importare un volume esistente in Trident.

```
tridentctl import volume <backendName> <volumeName> [flags]
```

Alias

`volume`, `v`

Flag

`-f`, `--filename string`: Percorso al file PVC YAML o JSON.
`-h`, `--help`: Guida per il volume.
`--no-manage`: Crea solo PV/PVC. Non presumere la gestione del ciclo di vita del volume.

installare

Utilizzare i `install` flag per installare Trident.

```
tridentctl install [flags]
```

Flag

`--autosupport-image string`: L'immagine del container per Autosupport Telemetry (predefinito "netapp/trident autosupport:<current-version>").
`--autosupport-proxy string`: L'indirizzo/porta di un proxy per l'invio di Autosupport Telemetry.
`--enable-node-prep`: Tenta di installare i pacchetti richiesti sui nodi.
`--generate-custom-yaml`: Genera file YAML senza installare nulla.
`-h, --help`: Guida per install.
`--http-request-timeout`: Ignora il timeout della richiesta HTTP per la REST API del controller Trident (predefinito 1m30s).
`--image-registry string`: L'indirizzo/porta di un registro di immagini interno.
`--k8s-timeout duration`: Il timeout per tutte le operazioni Kubernetes (predefinito 3m0s).
`--kubelet-dir string`: La posizione host dello stato interno di kubelet (predefinito "/var/lib/kubelet").
`--log-format string`: Il formato di logging di Trident (text, json) (predefinito "text").
`--node-prep`: Consente 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 di Trident supportata per questa funzionalità è 25.06.1.**
`--pv string`: Il nome del PV legacy utilizzato da Trident, assicurati che non esista (predefinito "trident").
`--pvc string`: Il nome del PVC legacy utilizzato da Trident, assicurati che non esista (predefinito "trident").
`--silence-autosupport`: Non inviare automaticamente i bundle di autosupport a NetApp (predefinito true).
`--silent`: Disabilita la maggior parte dell'output durante l'installazione.
`--trident-image string`: L'immagine Trident da installare.
`--k8s-api-qps`: Il limite di query per secondo (QPS) per le richieste API di Kubernetes (predefinito 100; opzionale).
`--use-custom-yaml`: Utilizza eventuali file YAML esistenti nella directory di setup.
`--use-ipv6`: Utilizza IPv6 per la comunicazione di Trident.

registri

Utilizzare `logs` flag per stampare i log da Trident.

```
tridentctl logs [flags]
```

Flag

`-a, --archive`: Crea un archivio di supporto con tutti i log salvo diversa indicazione.
`-h, --help`: Guida per i log.
`-l, --log string`: Trident log da visualizzare. Uno tra trident|auto|trident-operator|all (predefinito "auto").
`--node string`: Il nome del nodo Kubernetes da cui raccogliere i log dei pod del nodo.
`-p, --previous`: Ottieni i log per l'istanza precedente del container, se esistente.
`--sidecars`: Ottieni i log per i container sidecar.

Invia

Utilizzare il comando `send` per inviare una risorsa da Trident.

```
tridentctl send [option]
```

Opzioni

`autosupport`: Invia un archivio Autosupport a NetApp.

disinstallare

Utilizzare i `uninstall` flag per disinstallare Trident.

```
tridentctl uninstall [flags]
```

Flag

`-h, --help`: Guida per la disinstallazione.
`--silent`: Disattiva la maggior parte dell'output durante la disinstallazione.

aggiornamento

Utilizza il `update` comando per modificare una risorsa in Trident.

```
tridentctl update [option]
```

Opzioni

`backend`: Aggiorna un backend in Trident.

aggiorna lo stato del backend

Utilizza il comando `update backend state` 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), il backend non può essere aggiornato utilizzando un `backend.json` file.
- Se `userState` è stato impostato in un tbc, non può essere modificato utilizzando il comando `tridentctl update backend state <backend-name> --user-state suspended/normal`.
- Per ripristinare la possibilità di impostare il `userState` tramite tridentctl dopo che è stato impostato tramite tbc, il campo `userState` deve essere rimosso dal tbc. Questo può essere fatto usando il comando `kubectl edit tbc`. Dopo che il campo `userState` è stato rimosso, è possibile usare il comando `tridentctl update backend state` per modificare il `userState` di un backend.
- Utilizza il `tridentctl update backend state` per modificare il `userState`. Puoi anche aggiornare il `userState` utilizzando `TridentBackendConfig` o `backend.json` file; ciò innesca una reinizializzazione completa del backend e può richiedere molto tempo.

  #### Flag

  `-h, --help`: Aiuto per lo stato del backend.
  `--user-state`: Impostare su `suspended` per mettere in pausa le operazioni del backend. Impostare su `normal` per riprendere le operazioni del backend. Quando impostato su `suspended`:

- `AddVolume` and `Import Volume` sono in pausa.
- `CloneVolume`, `ResizeVolume`, `PublishVolume`, `UnPublishVolume`, `CreateSnapshot`, `GetSnapshot`, `RestoreSnapshot`, `DeleteSnapshot`, `RemoveVolume`, `GetVolumeExternal`, `ReconcileNodeAccess` rimangono disponibili.

È anche possibile aggiornare lo stato del backend utilizzando `userState` il campo nel file di configurazione del backend `TridentBackendConfig` o `backend.json`. Per ulteriori informazioni, fare riferimento a

"Opzioni per la gestione dei backend" e "Esegui la gestione del backend con kubectl".

Esempio:

JSON

Segui questi passaggi per aggiornare il `userState` utilizzando il `backend.json` file:

1. Modifica il `backend.json` file per includere il `userState` campo con il valore impostato su 'suspended'.
2. Aggiorna il backend utilizzando il `tridentctl update backend` comando e il percorso del file `backend.json` aggiornato.

   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 comando `kubectl edit <tbc-name> -n <namespace>`. Il seguente esempio aggiorna lo stato del backend in sospensione utilizzando l'opzione `userState: suspended`:

```
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` i flag per stampare la versione di `tridentctl` e il servizio Trident in esecuzione.

```
tridentctl version [flags]
```

Flag

`--client`: Solo versione client (nessun server richiesto).
`-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 d'ambiente PATH. Tutti i plugin rilevati sono elencati nella sezione plugin della guida di tridentctl. Facoltativamente, puoi anche limitare la ricerca specificando una cartella di plugin nella variabile d'ambiente TRIDENTCTL_PLUGIN_PATH (Example: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). Se la variabile viene utilizzata, tridentctl cerca solo nella cartella specificata.

Monitorare Trident

Trident fornisce una serie di endpoint di metriche Prometheus che puoi utilizzare per monitorare le prestazioni di Trident.

Panoramica

Le metriche fornite da Trident consentono di fare quanto segue:

- Tieni d'occhio lo stato di salute e la configurazione di Trident. Puoi verificare il successo delle operazioni e se riesce a comunicare con i backend come previsto.
- Esaminare le informazioni sull'utilizzo del backend e comprendere quanti volumi sono provisionati su un backend, la quantità di spazio consumata e così via.
- Mantieni una mappatura della quantità di volumi forniti sui backend disponibili.
- Monitora le prestazioni. Puoi osservare quanto tempo impiega Trident a comunicare con i backend ed eseguire operazioni.

NOTA	Per impostazione predefinita, le metriche di Trident sono esposte sulla porta di destinazione `8001` all'endpoint `/metrics`. Queste metriche sono **abilitate per impostazione predefinita** quando Trident è installato. È possibile configurare il consumo delle metriche di Trident anche tramite HTTPS sulla porta `8444`.

Cosa ti servirà

- Un cluster Kubernetes con Trident installato.
- Un'istanza di Prometheus. Questo può essere un "distribuzione containerizzata di Prometheus" oppure puoi scegliere di eseguire Prometheus come un "applicazione nativa".

Fase 1: definire un target Prometheus

Dovresti definire un target Prometheus per raccogliere le metriche e ottenere informazioni sui backend gestiti

da Trident, sui volumi che crea e così via. Vedere "Documentazione Prometheus Operator".

Fase 2: crea un Prometheus ServiceMonitor

Per consumare le metriche Trident, è necessario creare un Prometheus ServiceMonitor che monitora il `trident-csi` servizio e ascolta sulla porta `metrics`. Un esempio di ServiceMonitor è il 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 dal `trident-csi` servizio e cerca specificamente l'endpoint `metrics` 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 delle metriche. Kubelet può fornire informazioni sui volumi collegati, sui pod e su altre operazioni interne che gestisce. Consulta "qui".

Utilizza le metriche di Trident tramite HTTPS

Per utilizzare le metriche Trident tramite HTTPS (porta 8444), è necessario modificare la definizione ServiceMonitor per includere la configurazione TLS. È inoltre necessario copiare il `trident-csi` secret dallo `trident` namespace allo namespace in cui è in esecuzione Prometheus. È possibile farlo utilizzando il seguente comando:

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

Un esempio di ServiceMonitor per metriche 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 metriche HTTPS in tutti i metodi di installazione: tridentctl, Helm chart e Operator:

- Se si utilizza il `tridentctl install` comando, è possibile passare il `--https-metrics` flag per abilitare le metriche HTTPS.
- Se si utilizza il grafico Helm, è possibile impostare il parametro `httpsMetrics` per abilitare le metriche HTTPS.
- Se si utilizzano file YAML, è possibile aggiungere il `--https_metrics` flag al `trident-main` container nel `trident-deployment.yaml` file.

Passaggio 3: interrogare le metriche di Trident con PromQL

PromQL è utile per creare espressioni che restituiscono dati in serie temporali o tabulari.

Ecco alcune query PromQL che puoi utilizzare:

Ottieni informazioni sullo stato di 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 REST 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"})
```

Ottieni 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 del volume individuale

NOTA Questa opzione è abilitata solo se vengono raccolte anche le metriche kubelet.

- **Percentuale di spazio utilizzato per ciascun volume**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Scopri la telemetria Trident AutoSupport

Per impostazione predefinita, Trident invia metriche Prometheus e le informazioni di base del backend a NetApp su base giornaliera.

- Per impedire a Trident di inviare metriche Prometheus e informazioni di base sul backend a NetApp, passare il `--silence-autosupport` flag durante l'installazione di Trident.
- Trident può anche inviare i log dei container a NetApp Support su richiesta tramite `tridentctl send autosupport`. Sarà necessario attivare Trident per caricare i suoi log. Prima di inviare i log, è necessario accettare le NetApp "informativa sulla privacy".
- Se non diversamente specificato, Trident recupera i log delle ultime 24 ore.
- È possibile specificare il periodo di conservazione del log con il `--since` flag. Ad esempio: `tridentctl send autosupport --since=1h`. Queste informazioni vengono raccolte e inviate tramite un `trident-autosupport` container che viene installato insieme a Trident. È possibile ottenere l'immagine del container su "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 si presenta così:

```
---
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 stai utilizzando un registro privato per memorizzare le immagini container, puoi utilizzare il flag `--image-registry`.
- È anche possibile configurare gli URL proxy generando i file YAML di installazione. Questo può essere fatto utilizzando `tridentctl install --generate-custom-yaml` per creare i file YAML e aggiungendo l' `--proxy-url` argomento per il `trident-autosupport` container in `trident-`

`deployment.yaml`.

Disabilita metriche Trident

Per **disabilitare** le metriche dalla segnalazione, è necessario generare file YAML personalizzati (utilizzando il `--generate-custom-yaml` flag) e modificarli per rimuovere il `--metrics` flag dall'essere invocato per il `trident-main` container.

Disinstalla Trident

Dovresti usare lo stesso metodo per disinstallare Trident che hai usato per installare Trident.

Informazioni su questa attività

- Se hai bisogno di una correzione per bug riscontrati dopo un aggiornamento, problemi di dipendenza o un aggiornamento non riuscito o incompleto, dovresti disinstallare Trident e reinstallare la versione precedente utilizzando le istruzioni specifiche per quella "versione". Questo è l'unico metodo consigliato per effettuare il downgrade a una versione precedente.
- Per facilitare 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 suoi dati, fare riferimento a "Rimuovere completamente Trident e CRDs".

Prima di iniziare

Se si stanno dismettendo i cluster Kubernetes, è necessario eliminare tutte le applicazioni che utilizzano volumi creati da Trident prima della disinstallazione. Questo garantisce che le PVC siano annullate sui nodi Kubernetes prima di essere eliminate.

Determinare il metodo di installazione originale

Dovresti usare lo stesso metodo per disinstallare Trident che hai usato per installarlo. Prima di disinstallare, verifica quale versione hai usato per installare originariamente Trident.

1. Usa `kubectl get pods -n trident` per esaminare i pod.
   - Se non è presente alcun pod operatore, Trident è stato installato usando `tridentctl`.
   - Se è presente un pod operatore, Trident è stato installato utilizzando l'operatore Trident manualmente o tramite Helm.
2. Se è presente un operator pod, usa `kubectl describe tproc trident` per determinare se Trident è stato installato usando 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

Puoi disinstallare un'installazione dell'operatore Trident manualmente o usando Helm.

Disinstalla installazione manuale

Se hai installato Trident usando l'operatore, puoi disinstallarlo eseguendo una delle seguenti operazioni:

1. **Modifica `TridentOrchestrator` CR e imposta 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. Dovresti pulire il TridentOrchestrator e crearne uno nuovo se vuoi installare nuovamente Trident.

2. **Elimina `TridentOrchestrator`**: Rimuovendo la `TridentOrchestrator` CR utilizzata per distribuire Trident, si chiede all'operatore di disinstallare Trident. L'operatore elabora la rimozione di `TridentOrchestrator` e procede a rimuovere la distribuzione e il daemonset di Trident, eliminando i pod Trident creati durante l'installazione.

   ```
   kubectl delete -f deploy/<bundle.yaml> -n <namespace>
   ```

Disinstallare l'installazione di Helm

Se hai installato Trident tramite Helm, puoi disinstallarlo usando `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
```

Disinstalla 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 la distribuzione

È necessario installare e configurare i prerequisiti di protocollo necessari sul proprio host prima di poter distribuire Trident.

Verifica i requisiti

- Verifica che l'installazione soddisfi tutti i "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	Riavvia i nodi worker dopo aver installato gli strumenti NFS per prevenire errori durante il collegamento dei volumi ai container.

Strumenti iSCSI

Installa gli strumenti iSCSI utilizzando i comandi per il tuo 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. Imposta la scansione su manuale:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Abilita il multipathing:

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

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

5. Assicurarsi che `iscsid` e `multipathd` siano in funzione:

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

6. Abilita e avvia `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 di open-iscsi sia 2.0.874-5ubuntu2.10 o successiva (per bionic) o 2.0.874-7.1ubuntu6.1 o successiva (per focal):

```
dpkg -l open-iscsi
```

3. Imposta la scansione su manuale:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Abilita 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` che contenga `find_multipaths no` sotto `defaults`.

5. Assicurarsi che `open-iscsi` e `multipath-tools` siano abilitati e funzionanti:

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

Strumenti NVMe

Installare gli strumenti NVMe utilizzando i comandi per il proprio sistema operativo.

NOTA

- NVMe richiede RHEL 9 o versioni successive.
- Se la versione del kernel del nodo Kubernetes è troppo vecchia o se il pacchetto NVMe non è disponibile per la versione del kernel, potrebbe essere necessario aggiornare la versione del kernel del nodo a una 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 worker che eseguono RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) con PV FC, specificare `discard` mountOption nel StorageClass per eseguire la space reclamation in linea. Consulta "Documentazione Red Hat".

RHEL 8+

1. Installare i seguenti pacchetti di sistema:

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

2. Abilita il multipathing:

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

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

3. Assicurati 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. Abilita 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` che contenga `find_multipaths no` sotto `defaults`.

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

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

Distribuisci Trident

Trident per Docker offre un'integrazione diretta con l'ecosistema Docker per le piattaforme di storage NetApp. Supporta il provisioning e la gestione delle risorse di storage dalla piattaforma di storage agli host Docker, con un framework per l'aggiunta di ulteriori piattaforme in futuro.

Più istanze di Trident possono essere eseguite contemporaneamente sullo stesso host. Ciò consente connessioni simultanee a più sistemi di storage e tipi di storage, con la possibilità di personalizzare lo storage utilizzato per i volumi Docker.

Cosa ti servirà

Vedere il "prerequisiti per la distribuzione". Dopo aver verificato che i prerequisiti siano soddisfatti, si è pronti per distribuire Trident.

Metodo del plugin gestito da Docker (versione 1.13/17.03 e successive)

NOTA	**Prima di iniziare** Se hai utilizzato Trident prima di Docker 1.13/17.03 nel metodo daemon tradizionale, assicurati di arrestare il processo Trident e riavviare il daemon Docker prima di utilizzare il metodo del plugin gestito.

1. Arresta tutte le istanze in esecuzione:

```
pkill /usr/local/bin/netappdvp
pkill /usr/local/bin/trident
```

2. Riavvia Docker.

```
systemctl restart docker
```

3. Assicurati di avere installato Docker Engine 17.03 (new 1.13) o versione successiva.

```
docker --version
```

   Se la tua versione non è aggiornata "installa o aggiorna la tua installazione".

Passaggi

1. Crea un file di configurazione e specifica le opzioni come segue:
   - `config`: Il nome file predefinito è `config.json`, tuttavia è possibile utilizzare qualsiasi nome si scelga specificando l'opzione `config` con il nome del file. Il file di configurazione deve trovarsi nella directory `/etc/netappdvp` sul sistema host.
   - `log-level`: Specificare il livello di registrazione (`debug`, `info`, `warn`, `error`, `fatal`). Il valore predefinito è `info`.

- `debug`: Specifica se la registrazione del debug è abilitata. Il valore predefinito è false. Se true, sovrascrive log-level.

  i. Crea una posizione per il file di configurazione:

  ```
  sudo mkdir -p /etc/netappdvp
  ```

  ii. Crea 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. Avvia Trident utilizzando il sistema di plugin gestiti. Sostituisci `<version>` con la versione del plugin (xxx.xx.x) che stai utilizzando.

   ```
   docker plugin install --grant-all-permissions --alias netapp
   netapp/trident-plugin:<version> config=myConfigFile.json
   ```

3. Inizia a utilizzare Trident per consumare lo storage dal sistema configurato.

   a. Crea un volume denominato "firstVolume":

   ```
   docker volume create -d netapp --name firstVolume
   ```

   b. Crea un volume predefinito quando il container si avvia:

   ```
   docker run --rm -it --volume-driver netapp --volume
   secondVolume:/my_vol alpine ash
   ```

   c. Rimuovi il volume "firstVolume":

```
docker volume rm firstVolume
```

Metodo tradizionale (versione 1.12 o precedente)

Prima di iniziare

1. Assicurati di avere Docker versione 1.10 o successiva.

   ```
   docker --version
   ```

   Se la tua versione è obsoleta, aggiorna l'installazione.

   ```
   curl -fsSL https://get.docker.com/ | sh
   ```

   O, "segui le istruzioni per la tua distribuzione".

2. Assicurati che NFS e/o iSCSI siano configurati per il tuo sistema.

Passaggi

1. Installa e configura il NetApp Docker Volume Plugin:
   a. Scarica e decomprimi l'applicazione:

      ```
      wget
      https://github.com/NetApp/trident/releases/download/10.0/trident-
      installer-26.02.0.tar.gz
      tar zxf trident-installer-26.02.0.tar.gz
      ```

   b. Spostati in una posizione nel percorso del bin:

      ```
      sudo mv trident-installer/extras/bin/trident /usr/local/bin/
      sudo chown root:root /usr/local/bin/trident
      sudo chmod 755 /usr/local/bin/trident
      ```

   c. Crea una posizione per il file di configurazione:

      ```
      sudo mkdir -p /etc/netappdvp
      ```

   d. Crea il file di configurazione:

      ```
      cat << EOF > /etc/netappdvp/ontap-nas.json
      ```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
EOF
```

2. Dopo aver posizionato il binario e creato il file di configurazione, avvia il daemon Trident utilizzando il file di configurazione desiderato.

```
sudo trident --config=/etc/netappdvp/ontap-nas.json
```

NOTA Se non specificato, il nome predefinito per il driver del volume è "netapp".

Dopo aver avviato il demone, puoi creare e gestire i volumi utilizzando l'interfaccia Docker CLI.

3. Crea un volume:

```
docker volume create -d netapp --name trident_1
```

4. Provisionare un volume Docker quando si avvia un container:

```
docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol
alpine ash
```

5. Rimuovi un volume Docker:

```
docker volume rm trident_1
```

```
docker volume rm trident_2
```

Avvia Trident all'avvio del sistema

Un file di unità di esempio per i sistemi basati su systemd si trova a `contrib/trident.service.example` nel Git repo. Per usare il file con RHEL, procedere come segue:

1. Copia il file nella posizione corretta.

   Dovresti usare nomi unici per i file delle unità se hai più di un'istanza in esecuzione.

   ```
   cp contrib/trident.service.example
   /usr/lib/systemd/system/trident.service
   ```

2. Modifica il file, cambia la descrizione (riga 2) in modo che corrisponda al nome del driver e il percorso del file di configurazione (riga 9) in modo che rifletta il tuo ambiente.
3. Ricarica systemd per applicare le modifiche:

   ```
   systemctl daemon-reload
   ```

4. Abilita il servizio.

   Questo nome varia a seconda del nome dato al file nella `/usr/lib/systemd/system` directory.

   ```
   systemctl enable trident
   ```

5. Avvia il servizio.

   ```
   systemctl start trident
   ```

6. Visualizza lo stato.

   ```
   systemctl status trident
   ```

NOTA Ogni volta che si modifica il file dell'unità, eseguire il comando `systemctl daemon-reload` affinché sia a conoscenza delle modifiche.

Aggiorna o disinstalla Trident

È possibile aggiornare Trident per Docker senza alcun impatto sui volumi in uso. Durante il processo di aggiornamento ci sarà un breve periodo in cui i comandi `docker volume` diretti al plugin non avranno successo e le applicazioni non potranno montare i volumi fino a quando il plugin non sarà di nuovo in funzione. Nella maggior parte dei casi si tratta di una questione di secondi.

Aggiornamento

Eseguire i passaggi seguenti per aggiornare Trident for Docker.

Passaggi

1. Elenca i volumi esistenti:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

2. Disattiva il plugin:

```
docker plugin disable -f netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Aggiorna il plugin:

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```

NOTA La versione 18.01 di Trident sostituisce l'nDVP. Dovresti eseguire l'aggiornamento direttamente dall'immagine `netapp/ndvp-plugin` all'immagine `netapp/trident-plugin`.

4. Abilita il plugin:

```
docker plugin enable netapp:latest
```

5. Verificare che il plugin sia abilitato:

```
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 effettuando l'aggiornamento da una vecchia versione di Trident (pre-20.10) a Trident 20.10 o successiva, si potrebbe incorrere in un errore. Per ulteriori informazioni, fai riferimento a "Problemi noti". Se si incorre in questo errore, si dovrebbe prima disabilitare il plugin, poi rimuovere il plugin e quindi installare la versione richiesta di Trident passando un parametro di configurazione aggiuntivo: `docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all-permissions config=config.json`

Disinstallare

Eseguire i passaggi seguenti per disinstallare Trident for Docker.

Passaggi

1. Rimuovere tutti i volumi creati dal plugin.
2. Disattiva il plugin:

```
docker plugin disable netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Rimuovi il plugin:

```
docker plugin rm netapp:latest
```

Gestire i volumi

È possibile creare, clonare e rimuovere facilmente i volumi utilizzando i comandi standard `docker volume` con il nome del driver Trident specificato quando necessario.

Crea un volume

- Crea un volume con un driver utilizzando il nome predefinito:

```
docker volume create -d netapp --name firstVolume
```

- Crea un volume con una specifica istanza di Trident:

```
docker volume create -d ntap_bronze --name bronzeVolume
```

NOTA Se non si specifica alcun "opzioni", vengono utilizzati i valori predefiniti per il driver.

- Sostituisci le dimensioni predefinite del volume. Vedi il seguente esempio per creare un volume da 20 GiB con un driver:

```
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 (esempio: 10G, 20GB, 3TiB). Se non vengono specificate unità, il valore predefinito è G. Le unità di dimensione possono essere espresse come potenze di 2 (B, KiB, MiB, GiB, TiB) o potenze di 10 (B, KB, MB, GB, TB). Le unità stenografiche utilizzano potenze di 2 (G = GiB, T = TiB, …).

Rimuovi un volume

- Rimuovi il volume come qualsiasi altro volume Docker:

```
docker volume rm firstVolume
```

IMPORTANTE Quando si utilizza il `solidfire-san` driver, l'esempio precedente elimina e cancella il volume.

Eseguire i passaggi seguenti per aggiornare Trident for Docker.

Clonare un volume

Quando si utilizzano i driver di storage `ontap-nas`, `ontap-san` e `solidfire-san`, Trident può clonare i volumi. Quando si utilizzano i `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.

- Ispeziona il volume per enumerare le istantanee:

```
docker volume inspect <volume_name>
```

- Crea un nuovo volume da un volume esistente. Questo comporterà la creazione di una nuova istantanea:

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume>
```

- Crea un nuovo volume da un'istantanea esistente su un volume. Questo non creerà 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
```

Accedere ai volumi creati esternamente

È possibile accedere ai dispositivi a blocchi creati esternamente (o ai loro cloni) dai container usando Trident **solo** se non hanno partizioni e se il loro filesystem è supportato da Trident (ad esempio: un `ext4`-formattato `/dev/sdc1` non sarà accessibile tramite Trident).

Opzioni di volume specifiche del driver

Ogni driver di archiviazione ha una serie di opzioni diverse, che puoi specificare al momento della creazione del volume per personalizzare il risultato. Vedi di seguito le opzioni che si applicano al tuo sistema storage configurato.

L'uso di queste opzioni durante l'operazione di creazione del volume è semplice. Fornire l'opzione e il valore usando l'operatore `-o` durante l'operazione CLI. Questi valori sovrascrivono qualsiasi valore equivalente dal file di configurazione JSON.

Opzioni del volume ONTAP

Le opzioni di creazione del volume per NFS, iSCSI e FC includono le seguenti:

Opzione	Descrizione
`size`	La dimensione del volume, predefinita su 1 GiB.
`spaceReserve`	Thin o thick provision del volume, impostazione predefinita: thin. I valori validi sono `none` (thin provisioned) e `volume` (thick provisioned).
`snapshotPolicy`	In questo modo si imposterà la policy di snapshot sul valore desiderato. L'impostazione predefinita è `none`, il che significa che non verranno create automaticamente snapshot per il volume. A meno che non sia stato modificato dall'amministratore dello storage, su tutti i sistemi ONTAP esiste una policy denominata "default" che crea e conserva sei snapshot orarie, due giornaliere e due settimanali. I dati conservati in una snapshot possono essere recuperati accedendo alla directory `.snapshot` in qualsiasi directory del volume.
`snapshotReserve`	In questo modo la riserva di snapshot verrà impostata sulla percentuale desiderata. L'impostazione predefinita è nessun valore, il che significa che ONTAP selezionerà il snapshotReserve (di solito 5%) se hai selezionato un snapshotPolicy, o 0% se il snapshotPolicy è none. Puoi impostare il valore predefinito di snapshotReserve nel file di configurazione per tutti i backend ONTAP, e puoi usarlo come opzione di creazione del volume per tutti i backend ONTAP tranne ontap-nas-economy.

Opzione	Descrizione
`splitOnClone`	Quando si clona un volume, questo farà sì che ONTAP divida immediatamente il clone dal suo genitore. Il valore predefinito è `false`. Alcuni casi d'uso per la clonazione di volumi sono meglio serviti dividendo il clone dal suo genitore immediatamente dopo la creazione, perché è improbabile che ci sia un'opportunità di efficienza di archiviazione. Ad esempio, la clonazione di un database vuoto può offrire un grande risparmio di tempo ma poco risparmio di archiviazione, quindi è meglio dividere il clone immediatamente.
`encryption`	Abilita 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 fornito in Trident sarà abilitato per NAE. Per ulteriori informazioni, fare riferimento a: "Come funziona Trident con NVE e NAE".
`tieringPolicy`	Imposta la policy di tiering da utilizzare per il volume. Questo decide se i dati vengono spostati nel cloud tier quando diventano inattivi (cold).

Le seguenti opzioni aggiuntive sono per NFS **solo**:

Opzione	Descrizione
`unixPermissions`	Questo controlla l'impostazione dei permessi per il volume stesso. Per impostazione predefinita, i permessi saranno impostati su `---rwxr-xr-x`, o in notazione numerica 0755, e `root` sarà il proprietario. Funzionerà sia il formato testo che quello numerico.
`snapshotDir`	Impostando questo su `true` renderà la directory `.snapshot` visibile ai client che accedono al volume. Il valore predefinito è `false`, il che significa che la visibilità della directory `.snapshot` è disabilitata per impostazione predefinita. Alcune immagini, ad esempio l'immagine ufficiale MySQL, non funzionano come previsto quando la directory `.snapshot` è visibile.
`exportPolicy`	Imposta la policy di esportazione da utilizzare per il volume. Il valore predefinito è `default`.

Opzione	Descrizione
`securityStyle`	Imposta lo stile di sicurezza da utilizzare per l'accesso al volume. Il valore predefinito è `unix`. I valori validi sono `unix` e `mixed`.

Le seguenti opzioni aggiuntive sono per iSCSI **solo**:

Opzione	Descrizione
`fileSystemType`	Imposta il file system utilizzato per formattare i volumi iSCSI. Il valore predefinito è `ext4`. I valori validi sono `ext3`, `ext4` e `xfs`.
`spaceAllocation`	Impostare questo su `false` disattiverà la funzione di allocazione dello spazio della LUN. Il valore predefinito è `true`, il che significa che ONTAP notifica all'host quando il volume ha esaurito lo spazio e la LUN nel volume non può accettare scritture. Questa opzione consente inoltre a ONTAP di recuperare automaticamente lo spazio quando l'host elimina i dati.

Esempi

Vedere gli esempi seguenti:

- 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
```

- Crea un volume che ha il bit setUID abilitato:

```
docker volume create -d netapp --name demo -o unixPermissions=4755
```

La dimensione minima del volume è 20 MiB.

Se la riserva di snapshot non è specificata e la snapshot policy è `none`, Trident utilizza una riserva di snapshot dello 0%.

- Crea un volume senza criteri di snapshot e senza riserva di snapshot:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- Crea un volume senza criteri di snapshot e con una riserva di snapshot personalizzata del 10%:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- Crea un volume con un criterio 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 di snapshot e accettare la riserva di snapshot predefinita di ONTAP (di solito 5%):

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy
```

Opzioni dei volumi del software Element

Le opzioni del software Element espongono le dimensioni e i criteri di qualità del servizio (QoS) associati al volume. Quando il volume viene creato, il criterio QoS ad esso associato viene specificato usando la `-o type=service_level` nomenclatura.

Il primo passo per definire un livello di servizio QoS con il driver Element è creare almeno un tipo e specificare gli IOPS minimi, massimi e di burst associati a un nome nel file di configurazione.

Altre opzioni di creazione del volume del software Element includono le seguenti:

Opzione	Descrizione
`size`	La dimensione del volume, per impostazione 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 esempio di file di configurazione con le definizioni 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 sopra, abbiamo 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
```

Raccogli i registri

È possibile raccogliere i log per aiutare nella risoluzione dei problemi. Il metodo utilizzato

per raccogliere i log varia in base a come si esegue il plugin Docker.

Raccogli i registri per la risoluzione dei problemi

Passaggi

1. Se si esegue Trident utilizzando il metodo di plugin gestito consigliato (ovvero, utilizzando `docker plugin` 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, puoi abilitare la registrazione di debug.

2. Per abilitare la registrazione del debug, installare il plugin con la registrazione del debug abilitata:

```
docker plugin install netapp/trident-plugin:<version> --alias <alias>
debug=true
```

   Oppure, abilita il debug logging quando il plugin è già installato:

```
docker plugin disable <plugin>
```

```
docker plugin set <plugin> debug=true
```

```
docker plugin enable <plugin>
```

3. Se si esegue il binario stesso sull'host, i log sono disponibili nella directory dell'host `/var/log/netappdvp`. Per abilitare la registrazione del debug, specificare `-debug` quando si esegue il plugin.

Suggerimenti generali per la risoluzione dei problemi

- Il problema più comune che i nuovi utenti riscontrano è un errore di configurazione che impedisce l'inizializzazione del plugin. Quando ciò accade, è probabile che venga visualizzato un messaggio come questo quando si tenta di installare o abilitare il plugin:

`Error response from daemon: dial unix /run/docker/plugins/<id>/netapp.sock: connect: no such file or directory`

Ciò significa che il plugin non è riuscito ad avviarsi. Fortunatamente, il plugin è stato sviluppato con una funzionalità di registrazione completa che dovrebbe aiutarti a diagnosticare la maggior parte dei problemi che potresti incontrare.

- In caso di problemi con il montaggio di un PV su un container, assicurarsi che `rpcbind`sia installato e in esecuzione. Utilizzare il gestore pacchetti richiesto per il sistema operativo host e verificare che `rpcbind`sia in esecuzione. È possibile verificare lo stato del servizio rpcbind eseguendo un `systemctl status rpcbind` o un suo equivalente.

Gestisci più istanze di Trident

Le istanze multiple di Trident sono necessarie quando si desidera avere più configurazioni di storage disponibili simultaneamente. La chiave per le istanze multiple è assegnare loro nomi diversi usando l' `--alias` opzione con il plugin containerizzato, o l' `--volume-driver` opzione quando si istanzia Trident sull'host.

Passaggi per il plugin gestito da Docker (versione 1.13/17.03 o successiva)

1. Avvia la prima istanza specificando un alias e il file di configurazione.

```
docker plugin install --grant-all-permissions --alias silver
netapp/trident-plugin:21.07 config=silver.json
```

2. Avvia la seconda istanza, specificando un alias diverso e il file di configurazione.

```
docker plugin install --grant-all-permissions --alias gold
netapp/trident-plugin:21.07 config=gold.json
```

3. Crea 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
```

Passaggi per il tradizionale (versione 1.12 o precedente)

1. Avvia il plugin con una configurazione NFS utilizzando un ID driver personalizzato:

```
sudo trident --volume-driver=netapp-nas --config=/path/to/config
-nfs.json
```

2. Avvia il plugin con una configurazione iSCSI utilizzando un driver ID personalizzato:

```
sudo trident --volume-driver=netapp-san --config=/path/to/config
-iscsi.json
```

3. Effettua il 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

Visualizza le opzioni di configurazione disponibili per le configurazioni di Trident.

Opzioni di configurazione globale

Queste opzioni di configurazione si applicano a tutte le configurazioni Trident, indipendentemente dalla piattaforma di storage utilizzata.

Opzione	Descrizione	Esempio
`version`	Numero di versione del file di configurazione	`1`
`storageDriverName`	Nome del driver di storage	`ontap-nas`, `ontap-san`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `solidfire-san`
`storagePrefix`	Prefisso opzionale per i nomi dei volumi. Predefinito: `netappdvp_`.	`staging_`

Opzione	Descrizione	Esempio
`limitVolumeSize`	Restrizione opzionale sulle dimensioni dei volumi. Default: "" (non applicato)	`10g`

SUGGERIMENTO	Non utilizzare `storagePrefix` (compreso il valore predefinito) per i backend Element. Per impostazione predefinita, `solidfire-san` il driver ignorerà questa impostazione e non utilizzerà un prefisso. NetApp raccomanda di utilizzare un tenantID specifico per la mappatura dei volumi Docker o di utilizzare i dati dell'attributo che sono popolati con la versione di Docker, le informazioni sul driver e il nome raw di Docker nei casi in cui sia stata utilizzata una modifica del nome.

Le opzioni predefinite sono disponibili per evitare di doverle specificare su ogni volume creato. L'opzione `size` è disponibile per tutti i tipi di controller. Vedere la sezione di configurazione ONTAP per un esempio di come impostare la dimensione predefinita del volume.

Opzione	Descrizione	Esempio
`size`	Dimensione predefinita facoltativa per i nuovi volumi. Predefinita: `1G`	`10G`

Configurazione ONTAP

Oltre ai valori di configurazione globali sopra, 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 domain name (FQDN).	`10.0.0.1`

Opzione	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. Puoi specificare un fully-qualified domain name (FQDN) da usare per le operazioni di mount NFS, permettendoti di creare un DNS round-robin per bilanciare il carico tra più dataLIF. **ONTAP SAN drivers**: Non specificare per iSCSI o FC. Trident utilizza "ONTAP Selective LUN Map" per individuare i LIF iSCSI o FC necessari per stabilire una sessione multipath. Viene generato un avviso se `dataLIF` è definito esplicitamente.	`10.0.0.2`
`svm`	Macchina virtuale di storage da utilizzare (obbligatoria, se il LIF di gestione è un cluster LIF)	`svm_nfs`
`username`	Nome utente per connettersi al dispositivo di storage	`vsadmin`
`password`	Password per connettersi 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 FlexGroup volume.	`aggr1`
`limitAggregateUsage`	Facoltativo, il provisioning fallisce se l'utilizzo è superiore a questa percentuale	`75%`

Opzione	Descrizione	Esempio
`nfsMountOptions`	Controllo dettagliato delle opzioni di montaggio NFS; il valore predefinito è "-o nfsvers=3". **Disponibile solo per i `ontap-nas` e `ontap-nas-economy` driver**. "Vedi le informazioni sulla configurazione dell'host NFS qui".	`-o nfsvers=4`
`igroupName`	Trident crea e gestisce per-node `igroups` come `netappdvp`. Questo valore non può essere modificato o omesso. **Disponibile solo per il `ontap-san` driver**.	`netappdvp`
`limitVolumeSize`	Dimensione massima del volume richiedibile.	`300g`
`qtreesPerFlexvol`	Numero massimo di qtree per FlexVol, deve essere compreso nell'intervallo [50, 300], il valore predefinito è 200. **Per il `ontap-nas-economy` driver, questa opzione consente di personalizzare il numero massimo di qtree per FlexVol**.	`300`
`sanType`	**Supportato per `ontap-san` driver solo.** Utilizzare per selezionare `iscsi` per iSCSI, `nvme` per NVMe/TCP o `fcp` per SCSI over Fibre Channel (FC).	`iscsi` se vuoto
`limitVolumePoolSize`	**Supportato per `ontap-san-economy` e `ontap-san-economy` driver solo.** Limita le dimensioni di FlexVol nei driver ONTAP ontap-nas-economy e ontap-SAN-economy.	`300g`

Sono disponibili opzioni predefinite per evitare di doverle specificare su ogni volume che crei:

Opzione	Descrizione	Esempio
`spaceReserve`	Modalità di prenotazione dello spazio; `none` (thin provisioned) o `volume` (thick)	`none`

Opzione	Descrizione	Esempio
`snapshotPolicy`	policy di Snapshot da utilizzare, predefinita è `none`	`none`
`snapshotReserve`	Percentuale di riserva di Snapshot, il valore predefinito è "" per accettare il valore predefinito di ONTAP	`10`
`splitOnClone`	Dividi un clone dal suo genitore al momento della creazione, impostazione predefinita `false`	`false`
`encryption`	Abilita 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 fornito in Trident sarà abilitato per NAE. Per ulteriori informazioni, fare riferimento a: "Come funziona Trident con NVE e NAE".	true
`unixPermissions`	Opzione NAS per volumi NFS forniti, impostazione predefinita `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, predefinita su `default`	`default`
`securityStyle`	Opzione NAS per l'accesso al volume NFS fornito. NFS supporta `mixed` e `unix` stili di sicurezza. Il valore predefinito è `unix`.	`unix`
`fileSystemType`	Opzione SAN per selezionare il tipo di file system, il valore predefinito è `ext4`	`xfs`
`tieringPolicy`	Criterio di suddivisione in livelli da utilizzare, impostazione predefinita è `none`.	`none`

Opzione	Descrizione	Esempio
`skipRecovery Queue`	Durante l'eliminazione del volume, ignora la coda di ripristino nello storage ed elimina immediatamente il volume.	``

Opzioni di ridimensionamento

I `ontap-nas` e `ontap-san` driver creano un ONTAP FlexVol per ogni volume Docker. ONTAP supporta fino a 1000 FlexVols per nodo del cluster, con un massimo di 12.000 volumi FlexVol per cluster. Se i requisiti del volume Docker rientrano in questa limitazione, il driver `ontap-nas` è la soluzione NAS preferita grazie alle funzionalità aggiuntive offerte da FlexVols, come snapshot granulari a livello di volume Docker e clonazione.

Se hai bisogno di più volumi Docker di quanti ne possano essere gestiti dai limiti di FlexVol, scegli il `ontap-nas-economy` o il `ontap-san-economy` driver.

Il `ontap-nas-economy`driver crea volumi Docker come Qtree ONTAP all'interno di un pool di FlexVol volumi gestiti automaticamente. I Qtree offrono una scalabilità molto maggiore, fino a 100.000 per nodo del cluster e 2.400.000 per cluster, a scapito di alcune funzionalità. Il `ontap-nas-economy`driver non supporta snapshot o clonazioni granulari dei volumi Docker.

NOTA Il `ontap-nas-economy` driver non è attualmente supportato in Docker Swarm, perché Docker Swarm non orchestra la creazione di volumi su più nodi.

Il `ontap-san-economy` driver crea volumi Docker come LUN ONTAP all'interno di un pool condiviso di FlexVol volumi gestiti automaticamente. In questo modo, ogni FlexVol non è limitato a una sola LUN e offre una migliore scalabilità per i carichi di lavoro SAN. A seconda dell'array di storage, ONTAP supporta fino a 16384 LUN per cluster. Poiché i volumi sono LUN sottostanti, questo driver supporta snapshot e clonazione granulari dei volumi Docker.

Scegli il `ontap-nas-flexgroup` driver per aumentare il parallelismo su un singolo volume che può crescere fino a raggiungere l'ordine dei petabyte con miliardi di file. Alcuni casi d'uso 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 FlexGroup in Trident prevede inoltre le seguenti considerazioni:

- Richiede versione di ONTAP 9.2 o superiore.
- Al momento in cui scrivo, FlexGroups supporta solo NFS v3.
- Si consiglia di abilitare gli identificatori NFSv3 a 64 bit per la SVM.
- La dimensione minima consigliata per membro/volume FlexGroup è 100 GiB.
- La clonazione non è supportata per i volumi FlexGroup.

Per informazioni sui FlexGroups e sui carichi di lavoro appropriati per i FlexGroups fare riferimento a "NetApp FlexGroup volume Guida alle migliori pratiche e all'implementazione".

Per ottenere funzionalità avanzate e su larga scala nello stesso ambiente, puoi eseguire più istanze del Docker Volume Plugin, con una che utilizza `ontap-nas` e un'altra che utilizza `ontap-nas-economy`.

Ruolo ONTAP personalizzato per Trident

È possibile creare un ruolo di cluster ONTAP con privilegi minimi in modo da non dover utilizzare il ruolo di amministratore ONTAP per eseguire operazioni in Trident. Quando si include il nome utente in una configurazione backend di Trident, Trident utilizza il ruolo di cluster ONTAP creato per eseguire le operazioni.

Fare riferimento a "Generatore di ruoli personalizzati Trident" per ulteriori informazioni sulla creazione di ruoli personalizzati Trident.

Utilizzo di ONTAP CLI

1. Crea un nuovo ruolo utilizzando il seguente comando:

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. Crea 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. Assegna 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

Eseguire i seguenti passaggi in ONTAP System Manager:

1. **Crea un ruolo personalizzato**:
   a. Per creare un ruolo personalizzato a livello di cluster, selezionare **Cluster > Settings**.

      (Oppure) Per creare un ruolo personalizzato a livello SVM, selezionare **Archiviazione > Storage VM > `required SVM`> Impostazioni > Utenti e ruoli**.

   b. Selezionare l'icona della freccia (→) accanto a **Users and Roles**.
   c. Seleziona **+Add** in **Roles**.
   d. Definisci le regole per il ruolo e fai clic su **Save**.
2. **Mappa il ruolo all'utente Trident**: + Esegui i seguenti passaggi nella pagina **Utenti e ruoli**:
   a. Selezionare l'icona Aggiungi **+** sotto **Utenti**.
   b. Selezionare il nome utente richiesto e selezionare un ruolo nel menu a discesa per **Role**.
   c. Fare clic su **Save**.

Per maggiori informazioni, consultare le seguenti pagine:

- "Ruoli personalizzati per l'amministrazione di ONTAP" o "Definisci ruoli personalizzati"
- "Lavorare con ruoli e utenti"

Esempi di file di configurazione ONTAP

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 <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"
}
```

Esempio NVMe/TCP 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 SCSI su FC per <code>ontap-san</code> driver

```
{
  "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 software Element

Oltre ai valori di configurazione globali, quando si utilizza il software Element (NetApp HCI/SolidFire), sono disponibili queste opzioni.

Opzione	Descrizione	Esempio
`Endpoint`	https://<login>:<password>@<mvip>/json-rpc/<element-version>	https://admin:admin@192.168.160.3/json-rpc/8.0
`SVIP`	Indirizzo IP e porta iSCSI	10.0.0.7:3260
`TenantName`	SolidFireF Tenant da utilizzare (creato se non trovato)	`docker`

Opzione	Descrizione	Esempio
`InitiatorIFace`	Specificare l'interfaccia quando si limita il traffico iSCSI all'interfaccia non predefinita	`default`
`Types`	Specifiche QoS	Vedi esempio sotto
`LegacyNamePrefix`	Prefisso per le installazioni aggiornate di Trident. Se hai utilizzato una versione di Trident precedente alla 1.3.2 e esegui un aggiornamento con volumi esistenti, dovrai impostare questo valore per accedere ai tuoi vecchi volumi che erano stati mappati tramite il metodo volume-name.	`netappdvp-`

Il `solidfire-san` 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 e limitazioni noti

Trova informazioni sui problemi noti e sulle limitazioni quando si utilizza Trident con Docker.

L'aggiornamento di Trident Docker Volume Plugin alla versione 20.10 e successive da versioni precedenti provoca un errore di aggiornamento con l'errore no such file or directory.

Soluzione alternativa

1. Disattiva il plugin.

```
docker plugin disable -f netapp:latest
```

2. Rimuovi il plugin.

```
docker plugin rm -f netapp:latest
```

3. Reinstallare il plugin fornendo il parametro extra `config`.

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant
-all-permissions config=config.json
```

I nomi dei volumi devono essere lunghi almeno 2 caratteri.

NOTA	Si tratta di una limitazione del client Docker. Il client interpreterà un nome a carattere singolo come un percorso di Windows. "Vedi bug 25773"

Docker Swarm ha alcuni comportamenti che impediscono a Trident di supportarlo con ogni combinazione di storage e driver.

- Attualmente Docker Swarm utilizza il nome del volume invece dell'ID del volume come identificatore univoco del volume.
- Le richieste di volume vengono inviate simultaneamente a ogni nodo di un cluster Swarm.
- I plugin di volume (incluso Trident) devono essere eseguiti in modo indipendente su ogni nodo di un cluster Swarm. A causa del modo in cui ONTAP funziona e di come i driver `ontap-nas` e `ontap-san` funzionano, sono gli unici che possono operare entro queste limitazioni.

Il resto dei driver è soggetto a problemi come le race condition che possono portare alla creazione di un gran numero di volumi per una singola richiesta senza un chiaro "winner"; ad esempio, Element ha una funzione che consente ai volumi di avere lo stesso nome ma ID diversi.

NetApp ha fornito feedback al team Docker, ma non ha alcuna indicazione su eventuali azioni future.

Se è in corso 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 in fase di provisioning.

Buone pratiche e raccomandazioni

Distribuzione

Utilizza le raccomandazioni elencate qui quando distribuisci Trident.

Distribuisci in uno spazio dei nomi dedicato

"Spazi dei nomi" forniscono separazione amministrativa tra diverse applicazioni e rappresentano 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 gli spazi dei nomi nel cluster Kubernetes e di conseguenza sfrutta un account di servizio con privilegi elevati.

Inoltre, l'accesso al pod Trident potrebbe consentire a un utente di accedere alle credenziali del sistema storage e ad altre informazioni sensibili. È importante assicurarsi che gli utenti dell'applicazione e le applicazioni di gestione non abbiano la possibilità di accedere alle definizioni degli oggetti Trident o ai pod stessi.

Utilizza le quote e i limiti di intervallo per controllare il consumo di storage

Kubernetes offre due funzionalità che, se combinate, forniscono un potente meccanismo per limitare il consumo di risorse da parte delle applicazioni. Il "meccanismo di storage quota" consente all'amministratore di implementare limiti di consumo di capacità e numero di oggetti, sia globali che specifici per classe di storage, su base per-namespace. Inoltre, l'utilizzo di un "limite di range" garantisce che le richieste PVC rientrino sia in un valore minimo che in un valore massimo prima che la richiesta venga inoltrata al provisioner.

Questi valori sono definiti per ogni namespace, il che significa che per ogni namespace devono essere definiti valori in linea con i relativi requisiti di risorse. Vedere qui per informazioni su "come sfruttare le quote".

Configurazione dello storage

Ogni piattaforma di storage nel portafoglio NetApp ha funzionalità uniche che avvantaggiano le 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 a un'altra, tuttavia, nella scelta della piattaforma è necessario tenere conto delle esigenze dell'applicazione e del team che gestisce il dispositivo.

È consigliabile seguire le best practice di base per il sistema operativo host con il protocollo che si sta utilizzando. Facoltativamente, si potrebbe valutare l'integrazione delle best practice applicative, ove disponibili, con le impostazioni di backend, storage class e PVC per ottimizzare lo storage per applicazioni specifiche.

ONTAP e Cloud Volumes ONTAP best practices

Scopri le best practice per la configurazione di ONTAP e Cloud Volumes ONTAP per Trident.

Le seguenti raccomandazioni sono linee guida per la configurazione di ONTAP per carichi di lavoro containerizzati, che consumano volumi forniti dinamicamente da Trident. Ciascuna dovrebbe essere considerata e valutata per l'adeguatezza al proprio ambiente.

Utilizzare SVM dedicati a Trident

Le Storage Virtual Machines (SVM) forniscono isolamento e separazione amministrativa tra i tenant su un sistema ONTAP. Dedicare una SVM alle applicazioni consente la delega dei privilegi e consente di applicare le best practice per limitare il consumo di risorse.

Sono disponibili diverse opzioni per la gestione della SVM:

- Fornire l'interfaccia di gestione del cluster nella configurazione backend, insieme alle credenziali appropriate, e specificare il nome SVM.
- Crea un'interfaccia di gestione dedicata per l'SVM utilizzando ONTAP System Manager o la CLI.
- Condividere il ruolo di gestione con un'interfaccia dati NFS.

In ogni caso, l'interfaccia dovrebbe essere in DNS e il nome DNS dovrebbe essere utilizzato durante la configurazione di Trident. Questo aiuta a facilitare alcuni scenari di DR, ad esempio SVM-DR senza l'uso della conservazione dell'identità di rete.

Non esiste una preferenza tra un LIF di gestione dedicato o condiviso per l'SVM, tuttavia, è necessario assicurarsi che le policy di sicurezza della rete siano allineate con l'approccio scelto. In ogni caso, il LIF di gestione dovrebbe essere accessibile tramite DNS per garantire la massima flessibilità qualora "SVM-DR" venga utilizzato in combinazione con Trident.

Limita i volumi totali

I sistemi storage ONTAP hanno un limite massimo di volumi totali, che varia in base alla versione del software e alla piattaforma hardware. Fare riferimento a "NetApp Hardware Universe" per la propria piattaforma specifica e versione di ONTAP per determinare i limiti esatti. Quando i volumi totali sono esauriti, le operazioni di provisioning falliscono non solo per Trident, ma per tutte le richieste di storage.

I driver di Trident `ontap-nas` e `ontap-san` effettuano il provisioning di un FlexVolume per ogni Kubernetes Persistent Volume (PV) creato. Il driver `ontap-nas-economy` crea circa un FlexVolume ogni 200 PV (configurabile tra 50 e 300). Il driver `ontap-san-economy` crea circa un FlexVolume ogni 100 PV (configurabile tra 50 e 200). Per impedire che Trident consumi tutti i volumi disponibili sul sistema storage, è necessario impostare un limite sulla SVM. Puoi farlo 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 del tuo ambiente:

- Il numero di volumi esistenti nel cluster ONTAP
- Il numero di volumi che si prevede di effettuare il provisioning al di fuori di Trident per altre applicazioni
- Il numero di volumi persistenti che si prevede saranno consumati dalle applicazioni Kubernetes

Il `max-volumes` valore rappresenta il totale dei volumi forniti su tutti i nodi nel cluster ONTAP, e non su un singolo nodo ONTAP. Di conseguenza, potresti incontrare alcune condizioni in cui un nodo del cluster ONTAP potrebbe avere molti più o meno volumi forniti da Trident rispetto a un altro nodo.

Ad esempio, un cluster ONTAP a due nodi ha la capacità di ospitare un massimo di 2000 volumi FlexVol. Impostare i volumi totali massimi a 1250 sembra molto ragionevole. Tuttavia, se solo "aggregati" di un nodo vengono assegnati alla SVM, o se gli aggregati assegnati da un nodo non possono essere utilizzati per il provisioning (ad esempio, a causa della capacità), allora l'altro nodo diventa la destinazione per tutti i volumi

provisionati da Trident. Questo significa che il limite di volume per quel nodo potrebbe essere raggiunto prima che venga raggiunto il valore `max-volumes`, con un impatto sia su Trident sia sulle altre operazioni sui volumi che utilizzano quel nodo. **Puoi evitare questa situazione assicurandoti che gli aggregati di ciascun nodo del cluster siano assegnati alla SVM utilizzata da Trident in numero uguale.**

Clonare un volume

NetApp Trident supporta la clonazione dei volumi quando si utilizzano i `ontap-nas`, `ontap-san` e `solidfire-san` driver di storage. Quando si utilizzano i `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 un PVC associato a un diverso StorageClass. Eseguire operazioni di clonazione all'interno dello stesso StorageClass per garantire la compatibilità ed evitare comportamenti imprevisti.

Limita la dimensione massima dei volumi creati da Trident

Per configurare la dimensione massima dei volumi che possono essere creati da Trident, utilizzare il `limitVolumeSize` parametro nella `backend.json` definizione.

Oltre a controllare le dimensioni del volume nello storage array, dovresti anche sfruttare le funzionalità di Kubernetes.

Limita la dimensione massima dei FlexVols creati da Trident

Per configurare la dimensione massima per i FlexVols utilizzati come pool per i driver ontap-san-economy e ontap-nas-economy, utilizzare il `limitVolumePoolSize` parametro nella `backend.json` definizione.

Configurare Trident per utilizzare CHAP bidirezionale

È possibile specificare i nomi utente e le password dell'initiator e del target CHAP nella definizione del backend e fare in modo che Trident abiliti CHAP sulla SVM. Utilizzando il parametro `useCHAP` nella configurazione del backend, Trident autentica le connessioni iSCSI per i backend ONTAP con CHAP.

Crea e utilizza una policy QoS SVM

L'utilizzo di una policy QoS ONTAP applicata alla SVM limita il numero di IOPS utilizzabili dai volumi provisionati da Trident. Questo aiuta a "prevenire un bullo" o un container fuori controllo dall'influenzare i carichi di lavoro esterni alla SVM di Trident.

È possibile creare una policy QoS per la SVM in pochi passaggi. Consultare la documentazione per la propria versione di ONTAP per le informazioni più accurate. L'esempio seguente crea una policy QoS che limita il totale di IOPS disponibili per la SVM a 5000.

```
# 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 valutare l'utilizzo di un QoS minimo per garantire una quantità di throughput ai carichi di lavoro containerizzati. Il QoS adattivo non è compatibile con una policy a livello di SVM.

Il numero di IOPS dedicati ai carichi di lavoro containerizzati dipende da molti aspetti. Tra le altre cose, questi includono:

- Altri carichi di lavoro che utilizzano l'array di storage. Se sono presenti altri carichi di lavoro, non correlati all'implementazione di Kubernetes, che utilizzano le risorse di storage, è necessario prestare attenzione per garantire che tali carichi di lavoro non subiscano accidentalmente impatti negativi.
- Carichi di lavoro previsti in esecuzione nei container. Se carichi di lavoro con elevati requisiti IOPS vengono eseguiti nei container, una policy QoS bassa si traduce in un'esperienza negativa.

È importante ricordare che una policy QoS assegnata a livello di SVM fa sì che tutti i volumi forniti alla SVM condividano lo stesso pool di IOPS. Se una, o un numero limitato, di applicazioni containerizzate ha un elevato requisito di IOPS, potrebbe diventare un problema per gli altri carichi di lavoro containerizzati. Se questo è il caso, potresti voler considerare l'utilizzo di un'automazione esterna per assegnare policy QoS per volume.

IMPORTANTE	Dovresti assegnare il gruppo di policy QoS all'SVM **solo** se la versione di ONTAP è precedente alla 9.8.

Crea gruppi di policy QoS per Trident

La qualità del servizio (QoS) garantisce che le prestazioni dei carichi di lavoro critici non siano degradate da carichi di lavoro concorrenti. I gruppi di policy QoS di ONTAP forniscono opzioni QoS per i volumi e consentono agli utenti di definire il throughput massimo per uno o più carichi di lavoro. Per ulteriori informazioni sulla QoS, fare riferimento a "Garantire il throughput con QoS". È possibile specificare gruppi di policy QoS nel backend o in un pool di storage, e questi vengono applicati a ciascun volume creato in quel pool o backend.

ONTAP dispone di due tipi di gruppi di policy QoS: tradizionali e adattivi. I gruppi di policy tradizionali forniscono un throughput massimo (o minimo, nelle versioni successive) in IOPS. Il QoS adattivo scala automaticamente il throughput in base alle dimensioni del carico di lavoro, mantenendo il rapporto tra IOPS e TB|GB al variare delle 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.

Considerare quanto segue quando si creano gruppi di policy QoS:

- Dovresti impostare la `qosPolicy` chiave nel `defaults` blocco della configurazione del backend. Vedi 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 policy per volume, in modo che ogni volume riceva l'intero throughput come specificato dal gruppo di policy. I gruppi di policy condivisi non sono supportati.

Per ulteriori informazioni sui gruppi di policy QoS, fare riferimento a "Riferimento ai comandi ONTAP".

Limita l'accesso alle risorse di storage ai membri del cluster Kubernetes

Limitare l'accesso ai volumi NFS, alle LUN iSCSI e alle LUN FC create da Trident è un componente fondamentale della strategia di sicurezza per la distribuzione Kubernetes. In questo modo si impedisce agli host che non fanno parte del cluster di accedere ai volumi e di modificare potenzialmente i dati in modo imprevisto.

È importante comprendere che gli spazi dei nomi rappresentano il confine logico per le risorse in Kubernetes. Il presupposto è che le risorse nello stesso spazio dei nomi possano essere condivise, tuttavia, cosa importante, non esiste alcuna funzionalità cross-namespace. Ciò significa che, sebbene i PV siano oggetti globali, quando associati a un PVC sono accessibili solo ai pod che si trovano nello stesso spazio dei nomi. **È fondamentale garantire che gli spazi dei nomi vengano utilizzati per fornire la separazione quando appropriato.**

La preoccupazione principale per la maggior parte delle organizzazioni in merito alla sicurezza dei dati in un contesto Kubernetes è che un processo in un container possa accedere a storage montato sull'host, ma non destinato al container. "Spazi dei nomi" sono progettati per prevenire questo tipo di compromissione. Tuttavia, esiste un'eccezione: i container privilegiati.

Un container privilegiato è un container che viene eseguito con autorizzazioni a livello di host notevolmente superiori al normale. Queste non vengono negate per impostazione predefinita, quindi assicurati di disabilitare la funzionalità utilizzando "politiche di sicurezza del pod".

Per i volumi in cui è richiesto l'accesso sia da Kubernetes che da host esterni, lo storage dovrebbe essere gestito in modo tradizionale, con il PV introdotto dall'amministratore e non gestito da Trident. Questo garantisce che il volume di storage venga distrutto solo quando sia Kubernetes che 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 esterni al cluster.

Per le distribuzioni che dispongono di nodi infrastrutturali 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 che sono distribuiti su tali nodi infrastrutturali (ad esempio, i servizi di Metriche e Logging di OpenShift), e per le applicazioni standard che sono distribuite su nodi non infrastrutturali.

Utilizzare una policy di esportazione dedicata

È necessario assicurarsi che esista una policy di esportazione per ogni 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 di nodi.

In alternativa, puoi anche creare manualmente una policy di esportazione e popolarla con una o più regole di esportazione che elaborano ogni richiesta di accesso al nodo:

- Utilizzare il comando ONTAP CLI `vserver export-policy create` per creare la policy di esportazione.
- Aggiungere regole alla policy di esportazione utilizzando il `vserver export-policy rule create` comando ONTAP CLI.

L'esecuzione di questi comandi consente di limitare quali nodi Kubernetes hanno accesso ai dati.

Disabilita `showmount` per l'applicazione SVM

La `showmount`funzionalità consente a un client NFS di interrogare l'SVM per un elenco delle esportazioni NFS disponibili. Un pod distribuito nel cluster Kubernetes può emettere il comando `showmount -e` contro la SVM e ricevere un elenco dei mount disponibili, inclusi quelli a cui non ha accesso. Sebbene questo, di per sé, non costituisca una compromissione della sicurezza, fornisce informazioni non necessarie che potrebbero aiutare un utente non autorizzato a connettersi a un'esportazione NFS.

È necessario disabilitare `showmount` utilizzando il comando CLI di ONTAP a livello SVM:

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

Best practice per SolidFire

Scopri le best practice per la configurazione dello storage SolidFire per Trident.

Crea account SolidFire

Ogni SolidFire account rappresenta un proprietario univoco del volume e riceve il proprio set di credenziali Challenge-Handshake Authentication Protocol (CHAP). Puoi accedere ai volumi assegnati a un account utilizzando il nome dell'account e le relative credenziali CHAP oppure tramite 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 Quality of Service (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 su base per-volume. Le prestazioni per ogni volume possono essere garantite impostando tre parametri configurabili che definiscono la QoS: Min IOPS, Max IOPS e Burst IOPS.

Ecco i possibili valori minimi, massimi e burst di IOPS per la dimensione del blocco da 4Kb.

Parametro IOPS	Definizione	Valore minimo	Valore predefinito	Valore massimo(4Kb)
IOPS minimi	Il livello garantito di prestazioni per un volume.	50	50	15000
IOPS massimi	Le prestazioni non supereranno questo limite.	50	15000	200.000
IOPS a raffica	IOPS massimi consentiti in uno scenario di burst breve.	50	15000	200.000

NOTA Sebbene i valori Max IOPS e Burst IOPS possano essere impostati fino a 200.000, le prestazioni massime reali di un volume sono limitate dall'utilizzo del cluster e dalle prestazioni 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 "Quality of Service SolidFire" per ulteriori informazioni su QoS e prestazioni.

Autenticazione SolidFire

Element supporta due metodi di autenticazione: CHAP e Volume Access Groups (VAG). CHAP utilizza il protocollo CHAP per autenticare l'host al backend. Volume Access Groups controlla l'accesso ai volumi che mette a disposizione. NetApp consiglia di utilizzare CHAP per l'autenticazione in quanto è più semplice e non ha limiti di scala.

NOTA Trident con il provisioner CSI avanzato supporta l'uso dell'autenticazione CHAP. I VAG devono essere utilizzati solo nella modalità operativa tradizionale non CSI.

L'autenticazione CHAP (verifica che l'initiator sia l'utente previsto del volume) è supportata solo con il controllo dell'accesso basato sull'account. Se si utilizza CHAP per l'autenticazione, sono disponibili due opzioni: CHAP unidirezionale e CHAP bidirezionale. La CHAP unidirezionale autentica l'accesso al volume utilizzando il nome dell'account SolidFire e il segreto dell'initiator. L'opzione CHAP bidirezionale offre il modo più sicuro di autenticare il volume, poiché il volume autentica l'host tramite il nome dell'account e il segreto dell'initiator, e quindi l'host autentica il volume tramite il nome dell'account e il segreto della destinazione.

Tuttavia, se non è possibile abilitare CHAP e sono necessari i VAG, crea il gruppo di accesso e aggiungi gli initiator host e i volumi al gruppo di accesso. Ogni IQN che aggiungi a un gruppo di accesso può accedere a ogni volume del gruppo con o senza autenticazione CHAP. Se l'iSCSI initiator è configurato per usare l'autenticazione CHAP, viene utilizzato il controllo di accesso basato sull'account. Se l'iSCSI initiator non è configurato per usare l'autenticazione CHAP, viene utilizzato il controllo di accesso Volume Access Group.

Dove trovare ulteriori informazioni?

Di seguito sono elencate alcune delle documentazioni relative alle best practice. Cerca la "Libreria NetApp" per le versioni più aggiornate.

ONTAP

- "NFS Guida alle migliori pratiche e all'implementazione"
- "Amministrazione SAN" (per iSCSI)
- "Configurazione rapida iSCSI per RHEL"

Software Element

- "Configurazione di SolidFire per Linux"

NetApp HCI

- "Prerequisiti per l'implementazione di NetApp HCI"
- "Accedere al NetApp Deployment Engine"

Informazioni sulle best practice applicative

- "Best practice per MySQL su ONTAP"
- "Le migliori pratiche 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 tuo team NetApp e utilizzare il "Libreria NetApp" per trovare la documentazione più aggiornata.

Integra Trident

Per integrare Trident, è necessario integrare i seguenti elementi di progettazione e architettura: selezione e distribuzione del driver, progettazione della classe di archiviazione, progettazione del pool virtuale, impatto del Persistent Volume Claim (PVC) sul provisioning dello storage, operazioni sui volumi e distribuzione dei servizi OpenShift tramite Trident.

Selezione e distribuzione del driver

Seleziona e distribuisci un driver backend per il tuo sistema storage.

Driver backend ONTAP

I driver backend ONTAP si differenziano in base al protocollo utilizzato e al modo in cui i volumi vengono forniti sul sistema storage. Pertanto, valuta attentamente quale driver implementare.

A un livello superiore, se la tua applicazione ha componenti che necessitano di storage condiviso (più pod che accedono allo stesso PVC), i driver basati su NAS rappresentano la scelta predefinita, mentre i driver iSCSI basati su blocchi soddisfano le esigenze di 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, ci sono poche differenze tra loro per la maggior parte delle applicazioni, quindi spesso la decisione si basa sul fatto che sia necessario o meno uno storage condiviso (dove più di un pod avrà bisogno di accesso simultaneo).

I driver backend ONTAP disponibili sono:

- `ontap-nas`: Ogni PV fornito è un ONTAP FlexVolume 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`: Ogni PV è predisposto come un ONTAP FlexGroup completo e vengono utilizzati tutti gli aggregati assegnati a una SVM.
- `ontap-san`: Ogni PV fornito è una LUN all'interno del proprio FlexVolume.
- `ontap-san-economy`: Ogni PV fornito è una LUN, con un numero configurabile di LUN per FlexVolume (il valore predefinito è 100).

La scelta tra i tre driver NAS ha alcune ramificazioni sulle funzionalità rese disponibili all'applicazione.

Si noti che, nelle tabelle seguenti, non tutte le funzionalità sono esposte tramite Trident. Alcune devono essere applicate dall'amministratore dello storage dopo il provisioning se si desidera quella funzionalità. Le note a piè di pagina in apice distinguono la funzionalità per funzionalità e driver.

Driver NAS ONTAP	Istantanee	Cloni	Politiche di esportazione dinamiche	Multi-attach	QoS	Ridimensiona	Replica
`ontap-nas`	Sì	Sì	Sì [5]	Sì	Sì [1]	Sì	Sì [1]
`ontap-nas-economy`	NO [3]	NO [3]	Sì [5]	Sì	NO [3]	Sì	NO [3]
`ontap-nas-flexgroup`	Sì [1]	NO	Sì [5]	Sì	Sì [1]	Sì	Sì [1]

Trident offre 2 driver SAN per ONTAP, le cui funzionalità sono mostrate di seguito.

Driver SAN ONTAP	Istantanee	Cloni	Multi-attach	CHAP bidirezionale	QoS	Ridimensiona	Replica
`ontap-san`	Sì	Sì	Sì [4]	Sì	Sì [1]	Sì	Sì [1]
`ontap-san-economy`	Sì	Sì	Sì [4]	Sì	NO [3]	Sì	NO [3]

> Nota a piè di pagina per le tabelle precedenti: Sì [1]: Non gestito da Trident Sì [2]: Gestito da Trident, ma non PV granulare NO [3]: Non gestito da Trident e non PV granulare Sì [4]: Supportato per volumi raw-block Sì [5]: Supportato da Trident

Le funzionalità che non sono granulari al PV vengono applicate all'intero FlexVolume e tutti i PV (cioè qtrees o LUN in FlexVols condivisi) condivideranno una pianificazione comune.

Come si può vedere nelle tabelle precedenti, gran parte della funzionalità tra il `ontap-nas` e il `ontap-nas-economy` è la stessa. Tuttavia, poiché il driver `ontap-nas-economy` limita la possibilità di controllare la pianificazione a granularità per-PV, ciò può influire in particolare sulla pianificazione del disaster recovery e del backup. Per i team di sviluppo che desiderano sfruttare la funzionalità di clonazione PVC sullo storage ONTAP, ciò è possibile solo quando si utilizzano i driver `ontap-nas`, `ontap-san` o `ontap-san-economy`.

NOTA Il `solidfire-san` driver è anche in grado di clonare i PVC.

Driver di backend Cloud Volumes ONTAP

Cloud Volumes ONTAP fornisce il controllo dei dati insieme a funzionalità di storage di classe enterprise per vari casi d'uso, inclusi file share e storage a livello di blocco che servono protocolli NAS e SAN (NFS, SMB / CIFS e iSCSI). I driver compatibili per Cloud Volumes ONTAP sono `ontap-nas`, `ontap-nas-economy`, `ontap-san` e `ontap-san-economy`. Questi sono applicabili per Cloud Volumes ONTAP per Azure, Cloud Volumes ONTAP per GCP.

Driver di backend Amazon FSx for ONTAP

Amazon FSx for NetApp ONTAP consente di sfruttare le funzionalità, le prestazioni e le capacità amministrative di NetApp già note, sfruttando al contempo la semplicità, l'agilità, la sicurezza e la scalabilità dell'archiviazione dei dati su AWS. FSx for ONTAP supporta molte funzionalità del 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`.

NetApp HCI/SolidFire driver backend

Il `solidfire-san` driver utilizzato con le piattaforme NetApp HCI/SolidFire aiuta l'amministratore a configurare un backend Element per Trident sulla base di limiti QoS. Se si desidera progettare il backend per impostare limiti QoS specifici sui volumi forniti da Trident, utilizzare il parametro `type` nel file di backend. L'amministratore può anche limitare la dimensione del volume che può essere creato sullo storage utilizzando il parametro `limitVolumeSize`. Attualmente, le funzionalità di Element storage come il ridimensionamento del volume e la replica del volume non sono supportate tramite il `solidfire-san` driver. Queste operazioni devono essere eseguite manualmente tramite l'interfaccia web di Element Software.

Driver SolidFire	Istantanee	Cloni	Multi-attach	CHAP	QoS	Ridimensiona	Replica
`solidfire-san`	Sì	Sì	Sì [2]	Sì	Sì	Sì	Sì [1]

> Nota a piè di pagina: Sì [1]: Non gestito da Trident Sì [2]: Supportato per volumi raw-block

Driver backend di 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 del backend Trident per Azure NetApp Files".

Driver Azure NetApp Files	Istantanee	Cloni	Multi-attach	QoS	Espandi	Replica
`azure-netapp-files`	Sì	Sì	Sì	Sì	Sì	Sì [1]

> Nota a piè di pagina: Sì [1]: Non gestito da Trident

Progettazione della storage class

Le singole Storage class devono essere configurate e applicate per creare un oggetto Kubernetes Storage Class. Questa sezione illustra come progettare una storage class per la tua applicazione.

Utilizzo specifico del backend

Il filtraggio può essere utilizzato all'interno di uno specifico oggetto della storage class per determinare quale pool di storage o insieme di pool deve essere utilizzato con quella specifica storage class. Tre serie di filtri possono essere impostate nella Storage Class: `storagePools`, `additionalStoragePools`, e/o `excludeStoragePools`.

Il `storagePools` parametro consente di limitare lo storage all'insieme di pool che corrispondono agli attributi specificati. Il `additionalStoragePools` parametro viene utilizzato per estendere l'insieme di pool che Trident utilizza per il provisioning insieme all'insieme di pool selezionati dagli attributi e dai parametri `storagePools`. È possibile utilizzare uno dei due parametri da solo o entrambi insieme per assicurarsi che venga selezionato l'insieme appropriato di pool di storage.

Il `excludeStoragePools` parametro viene utilizzato per escludere specificamente l'insieme elencato di pool che corrispondono agli attributi.

Emula le policy QoS

Se si desidera progettare Storage Classes per emulare le politiche di Quality of Service, creare una Storage Class con l'attributo `media` come `hdd` o `ssd`. In base all'attributo `media` menzionato nella Storage Class, Trident selezionerà il backend appropriato che serve `hdd` o `ssd` aggregati per corrispondere all'attributo media e quindi dirigerà il provisioning dei volumi sull'aggregato specifico. Pertanto, è possibile creare una Storage Class PREMIUM con l'attributo `media` impostato come `ssd` che potrebbe essere classificata come la politica QoS PREMIUM. È possibile creare un'altra Storage Class STANDARD con l'attributo media impostato come `hdd' che potrebbe essere classificata come la politica QoS STANDARD. Si può anche utilizzare l'attributo ``IOPS'' nella Storage Class per reindirizzare il provisioning a un'appliance Element che può essere definita come una politica QoS.

Utilizzare il backend basato su funzionalità specifiche

Le classi di storage possono essere progettate per indirizzare il provisioning dei volumi su un backend specifico in cui sono abilitate funzionalità quali thin e thick provisioning, snapshot, cloni e crittografia. Per specificare quale storage utilizzare, crea Storage Classes che specificano il backend appropriato con la funzionalità richiesta abilitata.

Pool virtuali

I pool virtuali sono disponibili per tutti i backend Trident. È possibile definire pool virtuali per qualsiasi backend, utilizzando qualsiasi driver che Trident fornisce.

I pool virtuali consentono a un amministratore di creare un livello di astrazione sui backend che possono essere referenziati attraverso le Storage Classes, per una maggiore flessibilità e un posizionamento efficiente dei volumi sui backend. Backend diversi possono essere definiti con la stessa classe di servizio. Inoltre, è possibile creare più pool di storage sullo stesso backend ma con caratteristiche diverse. Quando una Storage Class è configurata con un selettore con etichette specifiche, Trident sceglie un backend che corrisponde a tutte le etichette del selettore per posizionare il volume. Se le etichette del selettore della Storage Class corrispondono a più pool di storage, Trident sceglierà uno di essi da cui effettuare il provisioning del volume.

Progettazione di pool virtuali

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 alleviato. Un pool virtuale è un'astrazione di livello introdotta tra il backend e la Kubernetes Storage Class, in modo che l'amministratore possa definire parametri insieme a etichette che possono essere referenziate tramite le Kubernetes Storage Classes come selettore, in modo backend-agnostic. I pool virtuali possono essere definiti per tutti i backend NetApp supportati con Trident. Questo elenco include SolidFire/NetApp HCI, ONTAP, così come Azure NetApp Files.

NOTA	Quando si definiscono 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 definire invece un nuovo pool virtuale.

Emulazione di diversi livelli di servizio/QoS

È possibile progettare pool virtuali per emulare classi di servizio. Utilizzando l'implementazione del pool virtuale per Cloud Volume Service for Azure NetApp Files, esaminiamo come possiamo configurare diverse classi di servizio. Configura il backend Azure NetApp Files con più etichette, che rappresentano diversi livelli di prestazioni. Imposta `servicelevel` l'aspetto sul livello di prestazioni appropriato e aggiungi altri aspetti richiesti sotto ciascuna etichetta. Ora crea diverse Kubernetes Storage Classes che verranno mappate a diversi pool virtuali. Utilizzando il campo `parameters.selector`, ogni StorageClass indica quali pool virtuali 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. Per farlo, configura il backend con più etichette e imposta gli aspetti richiesti sotto ciascuna etichetta. Ora crea diverse classi di storage Kubernetes utilizzando il campo `parameters.selector` che verranno mappate a diversi pool virtuali. I volumi che vengono provisionati sul backend avranno gli aspetti definiti nel pool virtuale scelto.

Caratteristiche del PVC che influenzano il provisioning dello storage

Alcuni parametri oltre la storage class richiesta possono influire sul processo decisionale di provisioning di Trident durante la creazione di un PVC.

Modalità di accesso

Quando si richiede 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 abbinare il protocollo storage utilizzato al metodo di accesso specificato in base alla seguente matrice. Questo è indipendente dalla piattaforma di storage sottostante.

	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
iSCSI	Sì	Sì	Sì (blocco raw)
NFS	Sì	Sì	Sì

Una richiesta per un ReadWriteMany PVC inviata a una distribuzione Trident senza un backend NFS configurato non comporterà il provisioning di alcun volume. Per questo motivo, il richiedente dovrebbe utilizzare la modalità di accesso appropriata per la propria applicazione.

Operazioni volume

Modificare i volumi persistenti

I volumi persistenti sono, con due eccezioni, oggetti immutabili in Kubernetes. Una volta creati, la reclaim policy e le dimensioni possono essere modificate. Tuttavia, ciò non impedisce che alcuni aspetti del volume vengano modificati al di fuori di Kubernetes. Questo può essere utile per personalizzare il volume per applicazioni specifiche, per garantire che la capacità non venga consumata accidentalmente o semplicemente per spostare il volume su un altro storage controller per qualsiasi motivo.

NOTA	I provisioner in-tree di Kubernetes al momento non supportano le operazioni di ridimensionamento dei volumi per PV NFS, iSCSI o FC. Trident supporta l'espansione dei volumi NFS, iSCSI e FC.

I dettagli di connessione del PV non possono essere modificati dopo la creazione.

Crea snapshot dei volumi su richiesta

Trident supporta la creazione di snapshot di volume on-demand e la creazione di PVC da snapshot utilizzando il framework CSI. Gli snapshot forniscono un metodo pratico per mantenere copie point-in-time dei dati e hanno un ciclo di vita indipendente dal PV sorgente in Kubernetes. Questi snapshot possono essere utilizzati per clonare i PVC.

Crea volumi da snapshot

Trident supporta anche la creazione di PersistentVolumes da snapshot di volume. Per farlo, è sufficiente creare un PersistentVolumeClaim e specificare il `datasource` come snapshot richiesto da cui deve essere creato il volume. Trident gestirà questo PVC creando un volume con i dati presenti nello snapshot. Con questa funzionalità, è possibile duplicare i dati tra regioni, creare ambienti di test, sostituire completamente un volume di produzione danneggiato o corrotto, oppure recuperare file e directory specifici e trasferirli su un altro volume collegato.

Sposta i volumi nel cluster

Gli amministratori di storage hanno la possibilità di spostare 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, purché l'aggregato di destinazione sia uno a cui l'SVM che Trident sta utilizzando ha accesso. È importante sottolineare che, se l'aggregato è stato appena aggiunto all'SVM, il backend dovrà essere aggiornato aggiungendolo nuovamente a Trident. Questo farà sì che Trident reinventari l'SVM in modo che il nuovo aggregato venga riconosciuto.

Tuttavia, lo spostamento di volumi tra backend non è supportato automaticamente da Trident. Questo include lo spostamento tra SVM nello stesso cluster, tra cluster o su una diversa piattaforma di storage (anche se tale sistema storage è connesso a Trident).

Se un volume viene copiato in un'altra posizione, la funzionalità di importazione del volume può essere utilizzata per importare i volumi correnti in Trident.

Espandi volumi

Trident supporta il ridimensionamento dei volumi permanenti 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 backend SolidFire/NetApp HCI. Per consentire un'eventuale espansione successiva, impostare `allowVolumeExpansion` su `true` nel StorageClass associato al volume. Ogni volta che è necessario ridimensionare il volume permanente, modificare l'annotazione `spec.resources.requests.storage` nella richiesta del volume permanente con la dimensione del volume richiesta. Trident si occuperà automaticamente del ridimensionamento del volume sul cluster.

Importa un volume esistente in Kubernetes

L'importazione di volumi offre la possibilità di importare un volume di storage esistente in un ambiente Kubernetes. Questa funzionalità è attualmente supportata dai `ontap-nas`, `ontap-nas-flexgroup`, `solidfire-san` e `azure-netapp-files` driver. Questa funzionalità è utile quando si esegue il porting di un'applicazione esistente in Kubernetes o durante scenari di disaster recovery.

Quando si utilizzano i driver ONTAP e `solidfire-san`, utilizzare il comando `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` per importare un volume esistente in Kubernetes da gestire tramite Trident. Il file PVC YAML o JSON utilizzato nel comando di importazione del volume punta a una storage class 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 funzionalità di importazione dei volumi possa distinguerli.

Se si utilizza il `azure-netapp-files` 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 univoco al volume.

Quando il comando sopra riportato viene eseguito, Trident troverà il volume sul backend e ne leggerà la dimensione. Aggiungerà automaticamente (e sovrascriverà se necessario) la dimensione del volume del PVC configurato. Trident crea quindi il nuovo PV e Kubernetes associa il PVC al PV.

Se un container è stato distribuito in modo da richiedere lo specifico PVC importato, rimarrà in sospeso finché la coppia PVC/PV non viene associata tramite il processo di importazione del volume. Dopo che la coppia PVC/PV è stata associata, il container dovrebbe avviarsi, a condizione che non ci siano altri problemi.

Servizio di registro

L'implementazione e la gestione dello storage per il registry sono state documentate su "netapp.io" nel "blog".

Servizio di logging

Come altri servizi OpenShift, il servizio di logging viene distribuito tramite Ansible con parametri di configurazione forniti dal file di inventario, ovvero hosts, forniti al playbook. Verranno trattati due metodi di installazione: la distribuzione del logging durante l'installazione iniziale di OpenShift e la distribuzione del

logging dopo che OpenShift è stato installato.

AVVERTENZA	A partire dalla versione 3.9 di Red Hat OpenShift, la documentazione ufficiale sconsiglia l'utilizzo di NFS per il servizio di logging a causa di problemi di corruzione dei dati. Questo si basa sui test effettuati da Red Hat sui propri prodotti. Il server NFS ONTAP non presenta questi problemi e può facilmente supportare un'implementazione di logging. In definitiva, la scelta del protocollo per il servizio di logging spetta a te, sappi solo che entrambi funzioneranno perfettamente utilizzando piattaforme NetApp e non c'è motivo di evitare NFS se questa è la tua preferenza.

Se si sceglie di utilizzare NFS con il servizio di registrazione, sarà necessario impostare la variabile Ansible `openshift_enable_unsupported_configurations` su `true` per impedire il fallimento del programma di installazione.

Inizia

Il servizio di logging può, facoltativamente, essere distribuito sia per le applicazioni sia per le operazioni principali del OpenShift cluster stesso. Se si sceglie di distribuire il logging 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", mentre l'istanza per le applicazioni no.

Configurare le variabili Ansible in base al metodo di distribuzione è importante per garantire che lo storage corretto sia utilizzato dai servizi sottostanti. Diamo un'occhiata alle 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 logging di Red Hat OpenShift" che dovrebbero essere riviste, configurate e utilizzate in base alla tua distribuzione.

Le variabili riportate nella tabella seguente porteranno il playbook di Ansible a creare un PV e un PVC per il servizio di logging utilizzando i dettagli forniti. Questo metodo è significativamente meno flessibile rispetto all'utilizzo del playbook di installazione dei componenti dopo l'installazione di OpenShift, tuttavia, se si dispone di volumi esistenti, è un'opzione.

Variabile	Dettagli
`openshift_logging_storage_kind`	Impostare `nfs` per fare in modo che il programma di installazione crei un PV NFS per il servizio di logging.
`openshift_logging_storage_host`	Il nome host o l'indirizzo IP dell'host NFS. Questo deve essere impostato sul dataLIF per la tua 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`, si dovrebbe usare quel 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`	La dimensione dell'esportazione NFS, ad esempio `100Gi`.

Se il OpenShift cluster è già in esecuzione e quindi Trident è stato distribuito e configurato, il programma di

installazione può utilizzare il provisioning dinamico per creare i volumi. Le seguenti variabili dovranno essere configurate.

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 storage class che verrà utilizzata nel PVC.
`openshift_logging_es_pvc_size`	La dimensione del volume richiesto nel PVC.
`openshift_logging_es_pvc_prefix`	Un prefisso per i PVC utilizzati dal servizio di logging.
`openshift_logging_es_ops_pvc_dynamic`	Impostare su `true` per utilizzare volumi forniti dinamicamente per l'istanza di logging ops.
`openshift_logging_es_ops_pvc_storage_class_name`	Il nome della storage class per l'istanza di ops logging.
`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 dell'istanza ops.

Distribuire lo stack di registrazione

Se si sta distribuendo il logging come parte del processo di installazione iniziale di OpenShift, è sufficiente seguire il processo di distribuzione standard. Ansible configurerà e distribuirà i servizi necessari e gli oggetti OpenShift in modo che il servizio sia disponibile non appena Ansible completa.

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 "Red Hat OpenShift Container Platform 3.11 documentazione" per la tua versione.

Servizio metriche

Il servizio metriche fornisce all'amministratore informazioni preziose sullo stato, l'utilizzo delle risorse e la disponibilità del OpenShift cluster. È inoltre necessario per la funzionalità di auto-scale del pod e molte organizzazioni utilizzano i dati del servizio metriche per le applicazioni di charge back e/o show back.

Come per il servizio di logging e OpenShift nel suo complesso, Ansible viene usato per distribuire il servizio metriche. Inoltre, come il servizio di logging, il servizio metriche può essere distribuito durante la configurazione iniziale del cluster o dopo che è operativo utilizzando il metodo di installazione dei componenti. Le tabelle seguenti contengono le variabili importanti quando si configura storage persistente per il servizio metriche.

NOTA	Le tabelle seguenti contengono solo le variabili rilevanti per la configurazione dello storage in relazione al servizio metriche. Ci sono molte altre opzioni che si trovano nella documentazione e che devono essere riviste, configurate e utilizzate in base alla propria distribuzione.

Variabile	Dettagli
`openshift_metrics_storage_kind`	Impostare `nfs` per fare in modo che il programma di installazione crei un PV NFS per il servizio di logging.

Variabile	Dettagli
`openshift_metrics_storage_host`	Il nome host o l'indirizzo IP dell'host NFS. Questo deve essere impostato sul dataLIF per il tuo SVM.
`openshift_metrics_storage_nfs_directory`	Il percorso di montaggio per l'esportazione NFS. Ad esempio, se il volume è giuntato come `/openshift_metrics`, si dovrebbe usare quel 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`	La dimensione dell'esportazione NFS, ad esempio `100Gi`.

Se il OpenShift cluster è già in esecuzione e quindi Trident è stato distribuito e configurato, il programma di installazione può utilizzare il provisioning dinamico per creare i volumi. Le seguenti variabili dovranno essere configurate.

Variabile	Dettagli
`openshift_metrics_cassandra_pvc_prefix`	Un prefisso da utilizzare per le metriche PVC.
`openshift_metrics_cassandra_pvc_size`	La dimensione dei volumi da richiedere.
`openshift_metrics_cassandra_storage_type`	Il tipo di storage da utilizzare per le metriche, deve essere impostato su dinamico affinché Ansible crei i PVC con la classe di storage appropriata.
`openshift_metrics_cassanda_pvc_storage_class_name`	Il nome della storage class da utilizzare.

Distribuire il servizio metriche

Con le variabili Ansible appropriate definite nel file hosts/inventory, distribuisci il servizio utilizzando Ansible. Se stai distribuendo al momento dell'installazione di OpenShift, allora il PV verrà creato e utilizzato automaticamente. Se stai distribuendo utilizzando i playbook dei componenti, dopo l'installazione di OpenShift, allora Ansible crea tutti i PVC necessari e, dopo che Trident ha eseguito il provisioning dello storage per essi, distribuisce 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 alla distribuzione di OpenShift di Red Hat" per la vostra versione affinché sia configurato per il vostro ambiente.

Protezione dei dati e disaster recovery

Scoprite le opzioni di protezione e recovery per Trident e i volumi creati utilizzando Trident. Dovreste avere una strategia di protezione e recovery 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 disastro.

Replica di Trident

Trident utilizza i CRD di Kubernetes per memorizzare e gestire il proprio stato e il cluster Kubernetes etcd per memorizzare i suoi metadati.

Passaggi

1. Eseguire il backup del cluster Kubernetes etcd utilizzando "Kubernetes: Backup di un cluster etcd".
2. Posiziona gli artefatti di backup su un volume FlexVol

NOTA	NetApp consiglia di proteggere l'SVM in cui risiede il FlexVol con una relazione di SnapMirror con un altro SVM.

Recovery di Trident

Utilizzando i CRD Kubernetes e lo snapshot etcd del cluster Kubernetes, è possibile recuperare Trident.

Passaggi

1. Dal SVM di destinazione, montare il volume che contiene i file di dati etcd di Kubernetes e i certificati sull'host che sarà configurato come nodo master.
2. Copia tutti i certificati richiesti relativi al cluster Kubernetes sotto `/etc/kubernetes/pki` e i file dei membri etcd sotto `/var/lib/etcd`.
3. Ripristina 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 di Trident siano state avviate e recuperare gli oggetti di Trident per verificare che tutti i dati siano disponibili.

Replica e recovery SVM

Trident non è in grado di configurare le relazioni di replica, tuttavia l'amministratore dello storage può utilizzare "ONTAP SnapMirror" per replicare un SVM.

In caso di disastro, è possibile attivare la SnapMirror destination SVM per iniziare a servire i dati. È possibile tornare al primario quando i sistemi vengono ripristinati.

Informazioni su questa attività

Considerare quanto segue quando si utilizza la funzionalità di replica SVM SnapMirror:

- Dovresti creare un backend distinto per ogni SVM con SVM-DR abilitato.
- Configura le classi di storage per selezionare i backend replicati solo quando necessario, per evitare che i volumi che non necessitano di replica vengano forniti sui backend che supportano SVM-DR.
- Gli amministratori delle applicazioni devono comprendere i costi aggiuntivi e la complessità associati alla replica e valutare attentamente il proprio piano di recovery prima di iniziare questo processo.

Replicazione SVM

È possibile utilizzare "ONTAP: SnapMirror SVM replicazione" per creare la relazione di replica SVM.

SnapMirror consente di impostare opzioni per controllare cosa replicare. Dovrai sapere quali opzioni hai selezionato quando esegui Recovery SVM tramite Trident.

- "-identity-preserve true" replica l'intera configurazione SVM.

- "-discard-configs network" esclude LIF e impostazioni di rete correlate.
- "-identity-preserve false" replica solo i volumi e la configurazione di sicurezza.

Recovery SVM tramite 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.

Passaggi

1. Annulla i trasferimenti SnapMirror pianificati e in corso, interrompi la relazione di replicazione, arresta l'SVM di origine e poi attiva l'SVM di destinazione SnapMirror.
2. Se hai specificato `-identity-preserve false` o `-discard-config network` durante la configurazione della replica SVM, aggiorna `managementLIF` e `dataLIF` nel file di definizione del backend Trident.
3. Conferma `storagePrefix` è presente nel file di definizione del backend Trident. Questo parametro non può essere modificato. Omettere `storagePrefix` causerà il fallimento dell'aggiornamento del backend.
4. Aggiorna 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 hai specificato `-identity-preserve false` o `discard-config network`, devi rimbalzare tutti i pod dell'applicazione.

> **NOTA** Se hai specificato `-identity-preserve true`, tutti i volumi forniti da Trident iniziano a servire dati quando la SVM di destinazione viene attivata.

Replicazione e recovery del volume

Trident non può configurare le relazioni di replica di SnapMirror, tuttavia l'amministratore dello storage può utilizzare "Replica e recovery ONTAP SnapMirror" per replicare i volumi creati da Trident.

È quindi possibile importare i volumi recuperati in Trident utilizzando "tridentctl volume import".

> **NOTA** L'importazione non è supportata sui `ontap-nas-economy`, `ontap-san-economy` o `ontap-flexgroup-economy` driver.

Protezione dei dati Snapshot

È possibile proteggere e ripristinare i dati utilizzando:

- Un controller di snapshot esterno e CRD per creare snapshot di volumi Kubernetes di Persistent Volumes (PV).

  "Istantanee del volume"

- ONTAP Snapshot per ripristinare l'intero contenuto di un volume o per recuperare singoli file o LUN.

"ONTAP Snapshot"

Automatizzare il failover delle applicazioni stateful con Trident

La funzionalità di force-detach di Trident consente di staccare automaticamente i volumi dai nodi non integri in un cluster Kubernetes, prevenendo la corruzione dei dati e garantendo la disponibilità delle applicazioni. Questa funzionalità è particolarmente utile negli scenari in cui i nodi diventano non responsivi o vengono messi offline per manutenzione.

Dettagli sul force detach

La disconnessione forzata è disponibile per `ontap-san`, `ontap-san-economy`, `ontap-nas` e `ontap-nas-economy` solo. Prima di abilitare la disconnessione forzata, è necessario abilitare l'arresto non regolare dei nodi (NGNS) sul cluster Kubernetes. NGNS è abilitato per impostazione predefinita per Kubernetes 1.28 e versioni successive. Per ulteriori informazioni, fai riferimento a "Kubernetes: arresto non corretto del nodo".

NOTA	Quando si utilizza il `ontap-nas` o `ontap-nas-economy` driver, è necessario impostare il parametro `autoExportPolicy` nella configurazione del backend su `true` in modo che Trident possa limitare l'accesso dal nodo Kubernetes con il taint applicato utilizzando policy di esportazione gestite.

ATTENZIONE	Poiché Trident si basa su Kubernetes NGNS, non rimuovere `out-of-service` le taint da un nodo non integro finché tutti i carichi di lavoro non tollerabili non vengono riprogrammati. L'applicazione o la rimozione sconsiderata delle taint può compromettere la protezione dei dati backend.

Quando l'amministratore del cluster Kubernetes ha applicato la `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` taint al nodo e `enableForceDetach` è impostato su `true`, Trident determinerà lo stato del nodo e:

1. Interrompi l'accesso I/O backend per i volumi montati su quel nodo.
2. Contrassegna l'oggetto nodo Trident come `dirty` (non sicuro per nuove pubblicazioni).

NOTA	Il controller Trident rifiuterà le nuove richieste di pubblicazione di volumi finché il nodo non verrà riqualificato (dopo essere stato contrassegnato come `dirty`) dal pod Trident del nodo. Qualsiasi carico di lavoro pianificato con un PVC montato (anche dopo che il nodo del cluster è integro e pronto) non verrà accettato finché Trident non potrà verificare il nodo `clean` (sicuro per nuove pubblicazioni).

Quando la salute del nodo viene ripristinata e la contaminazione viene rimossa, Trident:

1. Identificare e pulire i percorsi pubblicati obsoleti sul nodo.
2. Se il nodo si trova in uno stato `cleanable` (la contaminazione fuori servizio è stata rimossa e il nodo è in stato `Ready`) e tutti i percorsi pubblicati obsoleti sono puliti, Trident riammetterà il nodo come `clean` e consentirà nuovi volumi pubblicati sul nodo.

Dettagli sul failover automatico

È possibile automatizzare il processo di distacco forzato tramite integrazione con "operatore node health check (NHC)". Quando si verifica un errore di nodo, NHC attiva la Trident node remediation (TNR) e forza il distacco automaticamente creando una TridentNodeRemediation CR nello spazio dei nomi di Trident che definisce il nodo in errore. La TNR viene creata solo in caso di errore del nodo e rimossa 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 dirty, impedendo qualsiasi nuova pubblicazione di volumi e inizia a rimuovere i pod supportati dal force-detach e i relativi allegati ai volumi.

Tutti i volumi/PVC supportati da force-detach sono supportati da automated-failover:

- Volumi NAS e NAS-economy che utilizzano policy di auto-export (SMB non è ancora supportato).
- SAN e volumi SAN-economy.

Consulta Dettagli sul force detach.

Comportamento predefinito:

- I pod che utilizzano volumi supportati dal force-detach vengono rimossi dal nodo in errore. Kubernetes ripianificherà questi pod su un nodo funzionante.
- I pod che utilizzano un volume non supportato da force-detach, 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` sia impostata.

Sostituzione del 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. Applica le annotazioni alla specifica del pod di deployment/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 guasti per i volumi che supportano force-detach. • Per i volumi che non supportano il force-detach, esiste il rischio di corruzione dei dati e problemi di multi-attach.

CR TridentNodeRemediation

Il CR TridentNodeRemediation (TNR) definisce un nodo guasto. Il nome del TNR è il nome del nodo guasto.

Esempio TNR:

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

TNR states: 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*:
  - Interrompi 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 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 remediation e ripristino del nodo sono stati completati con successo. Il nodo è pulito e pronto per nuove pubblicazioni di volumi.
- *Non riuscito*:
  - Errore irreversibile. I motivi dell'errore sono impostati nel campo status.message della CR.

Abilitazione del failover automatico

Prerequisiti:

- Assicurarsi che la disconnessione forzata sia abilitata prima di abilitare il failover automatico. Per ulteriori informazioni, fai riferimento a Dettagli sul force detach.
- 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 Node Health check: `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.

Vedi "Operatore Node Health Check" per maggiori informazioni.

Passaggi

1. Crea una CR NodeHealthCheck (NHC) nel namespace 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. Applica il controllo di integrità del nodo CR nel namespace `trident`.

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

Il CR sopra indicato è configurato per monitorare i nodi worker di K8s per le condizioni Ready: false e Unknown. Il failover automatico verrà attivato quando un nodo passa allo stato Ready: false o Ready: Unknown.

Il `unhealthyConditions` nella CR utilizza un grace period 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 valore predefinito di 40 secondi di attesa dopo l'ultimo heartbeat prima di impostare Ready: false. Questo grace period può essere personalizzato nelle opzioni di deployment di K8s.

Per ulteriori opzioni di configurazione, fare riferimento a "Documentazione di Node-Healthcheck-Operator".

Informazioni aggiuntive sulla configurazione

Quando Trident viene installato con force-detach abilitato, 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 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: {}
```

ClusterRole:

Un ruolo cluster viene anche aggiunto durante l'installazione quando il force-detach è abilitato. Questo dà a NHC i permessi sui TNR nel namespace 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 K8s, quando è previsto che i nodi si fermino o si riavviino. È possibile sospendere il CR NHC (descritto sopra) applicando una patch al relativo CR:

```
kubectl patch NodeHealthCheck <cr-name> --patch
'{"spec":{"pauseRequests":["<description-for-reason-of-pause>"]}}' --type=merge
```

Questa operazione sospende il failover automatico. Per riattivare il failover automatico, rimuovere pauseRequests dalle specifiche dopo il completamento della manutenzione.

Limitazioni

- Le operazioni di I/O vengono impedite solo sui nodi non riusciti per i volumi supportati da force-detach. Solo i pod che utilizzano volumi/PVC supportati da force-detach vengono rimossi automaticamente.
- Automatic-failover e force-detach vengono eseguiti all'interno del pod trident-controller. Se il nodo che ospita trident-controller si guasta, l'automatic-failover sarà 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 dei guasti dei nodi per attivare il failover automatico. Per garantire la compatibilità con il meccanismo di failover automatizzato, la soluzione personalizzata dovrebbe:

- Crea un TNR quando viene rilevato un errore del nodo, utilizzando il nome del nodo non funzionante come nome CR del TNR.
- Eliminare il TNR quando il nodo è stato ripristinato e il TNR è nello stato Succeeded.

Sicurezza

Sicurezza

Utilizzare le raccomandazioni elencate qui per garantire che l'installazione di Trident sia sicura.

Esegui Trident nel suo namespace

È importante impedire alle applicazioni, agli amministratori delle applicazioni, agli utenti e alle applicazioni di gestione di accedere alle definizioni degli oggetti Trident o ai pod per garantire uno storage affidabile e bloccare potenziali attività dannose.

Per separare le altre applicazioni e gli altri utenti da Trident, installa sempre Trident nel proprio namespace Kubernetes (`trident`). Mettere Trident nel proprio namespace garantisce che solo il personale amministrativo di Kubernetes abbia accesso al pod Trident e agli artefatti (come backend e segreti CHAP, se applicabile) archiviati negli oggetti CRD con namespace. Devi assicurarti di consentire solo agli amministratori l'accesso al namespace Trident e quindi l'accesso all'applicazione `tridentctl`.

Utilizzare l'autenticazione CHAP con i backend ONTAP SAN

Trident supporta l'autenticazione basata su CHAP per i carichi di lavoro ONTAP SAN (utilizzando i `ontap-san`

e `ontap-san-economy` driver). NetApp consiglia di utilizzare CHAP bidirezionale con Trident per l'autenticazione tra un host e il backend di storage.

Per i backend ONTAP che utilizzano i driver di storage SAN, Trident può configurare 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 ONTAP SAN" per comprendere come Trident configura il CHAP sui backend ONTAP.

Utilizzare l'autenticazione CHAP con NetApp HCI e SolidFire backends

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 è installato, gestisce i segreti CHAP e li memorizza in un oggetto CR `tridentvolume` per il rispettivo PV. 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 che vengono creati da Trident non sono associati ad alcun Volume Access Group.

Usa Trident con NVE e NAE

NetApp ONTAP fornisce la crittografia dei dati a riposo per proteggere i dati sensibili in caso di furto, restituzione o riutilizzo di un disco. Per dettagli, consultare "Panoramica sulla configurazione della crittografia del volume NetApp".

- Se NAE è abilitato sul backend, qualsiasi volume fornito in Trident sarà NAE-enabled.
  - È possibile impostare il flag di crittografia NVE su `""` per creare volumi abilitati per NAE.
- Se NAE non è abilitato sul backend, qualsiasi volume fornito in Trident sarà abilitato per NVE, a meno che il flag di crittografia NVE non sia impostato su `false` (il valore predefinito) nella configurazione del backend.

NOTA

I volumi creati in Trident su un backend abilitato NAE devono essere crittografati NVE o NAE.

- È possibile impostare il flag di crittografia NVE su `true` nella configurazione del backend Trident per ignorare la crittografia NAE e utilizzare una chiave di crittografia specifica per ogni volume.
- Impostando il flag di crittografia NVE `false` su un backend abilitato per NAE si crea un volume abilitato per NAE. Non è possibile disabilitare la crittografia NAE impostando il flag di crittografia NVE a `false`.

- È possibile creare manualmente un volume NVE in Trident impostando esplicitamente il flag di crittografia NVE su `true`.

Per maggiori informazioni sulle opzioni di configurazione del backend, fare riferimento a:

- "Opzioni di configurazione SAN ONTAP"
- "Opzioni di configurazione NAS ONTAP"

Linux Unified Key Setup (LUKS)

È possibile abilitare Linux Unified Key Setup (LUKS) per crittografare i volumi ONTAP SAN e ONTAP SAN ECONOMY su Trident. Trident supporta la rotazione della

passphrase e l'espansione del volume per i volumi crittografati con LUKS.

In Trident, i volumi crittografati con LUKS utilizzano il cifrario e la modalità aes-xts-plain64, come raccomandato da "NIST".

NOTA	La crittografia LUKS non è supportata per i sistemi ASA r2. Per informazioni sui sistemi ASA r2, vedere "Scopri i sistemi di storage ASA r2".

Prima di iniziare

- I nodi worker devono avere installato cryptsetup 2.1 o superiore (ma inferiore a 3.0). Per ulteriori informazioni, visitare "Gitlab: cryptsetup".
- Per motivi di prestazioni, NetApp raccomanda che i nodi worker supportino Advanced Encryption Standard New Instructions (AES-NI). Per verificare il supporto di 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, visitare: "Intel: Advanced Encryption Standard Instructions (AES-NI)".

Abilita la crittografia LUKS

È possibile abilitare la crittografia per volume, lato host, utilizzando Linux Unified Key Setup (LUKS) per ONTAP SAN e ONTAP SAN ECONOMY volumi.

Passaggi

1. Definire gli attributi di crittografia LUKS nella configurazione backend. Per ulteriori informazioni sulle opzioni di configurazione backend 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 archiviazione 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. Crea un segreto che contenga la passphrase LUKS. Ad esempio:

```
kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA
```

Limitazioni

I volumi crittografati LUKS non possono sfruttare la deduplicazione e la compressione di ONTAP.

Configurazione backend per l'importazione di volumi LUKS

Per importare un volume LUKS, è necessario impostare `luksEncryption` su (`true` sul backend. L'opzione `luksEncryption` indica a Trident se il volume è LUKS-compliant (`true` o non LUKS-compliant (`false`, come mostrato 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 i volumi LUKS in modo dinamico, impostare l'annotazione `trident.netapp.io/luksEncryption` a `true` e includere una storage class abilitata per LUKS nel PVC come mostrato 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
```

Ruota una passphrase LUKS

È possibile ruotare la passphrase LUKS e confermare la rotazione.

ATTENZIONE	Non dimenticare una passphrase finché non hai verificato che non sia più referenziata da alcun volume, snapshot o segreto. Se una passphrase referenziata viene persa, potresti non essere in grado di montare il volume e i dati rimarranno crittografati e inaccessibili.

Informazioni su questa attività

La rotazione della passphrase LUKS avviene quando viene creato un pod che monta il volume dopo che è stata specificata una nuova passphrase LUKS. Quando viene creato un nuovo pod, Trident confronta la passphrase LUKS sul volume con la passphrase attiva nel secret.

- Se la passphrase sul volume non corrisponde alla passphrase attiva nel secret, si verifica la rotazione.
- Se la passphrase sul volume corrisponde alla passphrase attiva nel segreto, il `previous-luks-passphrase` parametro viene ignorato.

Passaggi

1. Aggiungere i `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 la nuova e la precedente passphrase. Assicurarsi che `previous-luke-passphrase-name` e `previous-luks-passphrase` corrispondano alla 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. Crea un nuovo pod che monta il volume. Questo è necessario 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 sul volume e sullo snapshot viene restituita solo la nuova passphrase.

NOTA	Se vengono restituite due passphrase, ad esempio `luksPassphraseNames: ["B", "A"]`, la rotazione è incompleta. Puoi attivare un nuovo pod per tentare di completare la rotazione.

Abilita l'espansione del volume

È possibile abilitare l'espansione del volume su un volume crittografato con LUKS.

Passaggi

1. Abilita la funzionalità `CSINodeExpandSecret` feature gate (beta 1.25+). Consultare "Kubernetes 1.25: utilizzare i Secrets per l'espansione dei volumi CSI basata sui nodi" per i dettagli.
2. Aggiungere i `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.

Crittografia in volo Kerberos

Utilizzando la crittografia Kerberos in volo, puoi migliorare la sicurezza dell'accesso ai dati abilitando la crittografia per il traffico tra il tuo cluster gestito e il backend di storage.

Trident supporta la crittografia Kerberos per ONTAP come storage backend:

- **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.

È possibile creare, eliminare, ridimensionare, creare snapshot, clonare, clonare in sola lettura e importare volumi che utilizzano NFS con crittografia.

Configura la crittografia Kerberos in volo con i volumi ONTAP on-premise

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 storage on-premise ONTAP è supportata solo utilizzando il `ontap-nas` storage driver.

Prima di iniziare

- Assicurati di avere accesso all' `tridentctl` utility.
- Assicurati di avere accesso come amministratore al backend di storage ONTAP.
- Assicurati di conoscere il nome del volume o dei volumi che condividerai dal backend di storage ONTAP.
- Assicurarsi di aver preparato la macchina virtuale di storage ONTAP per supportare la crittografia Kerberos per i volumi NFS. Consulta "Abilitare 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".

Aggiungi o modifica le policy di esportazione ONTAP

È necessario aggiungere regole alle policy di esportazione ONTAP esistenti o creare nuove policy di esportazione che supportino la crittografia Kerberos per il volume root della storage VM ONTAP, così come per qualsiasi volume ONTAP condiviso con il cluster Kubernetes upstream. Le regole delle policy di esportazione che aggiungi, o le nuove policy di esportazione che crei, devono supportare i seguenti protocolli di accesso e permessi 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 per il volume:

- **Kerberos 5** - (autenticazione e crittografia)
- **Kerberos 5i** - (autenticazione e crittografia con protezione dell'identità)
- **Kerberos 5p** - (autenticazione e crittografia con protezione dell'identità e della privacy)

Configurare la regola della policy di esportazione ONTAP con le autorizzazioni di accesso appropriate. Ad esempio, se i cluster monteranno i volumi NFS con un misto di crittografia Kerberos 5i e Kerberos 5p, utilizzare le seguenti impostazioni di accesso:

Tipo	Accesso in sola lettura	Accesso in lettura/scrittura	Accesso come superutente
UNIX	Abilitato	Abilitato	Abilitato
Kerberos 5i	Abilitato	Abilitato	Abilitato
Kerberos 5p	Abilitato	Abilitato	Abilitato

Consultare la seguente documentazione per informazioni su come creare le policy di esportazione ONTAP e le regole delle policy di esportazione:

- "Creare una policy di esportazione"
- "Aggiungere una regola a una policy di esportazione"

Creare un backend di storage

È possibile creare una configurazione del backend di storage Trident che includa la funzionalità di crittografia Kerberos.

Informazioni su questa attività

Quando si crea un file di configurazione del backend di archiviazione che configura la crittografia Kerberos, è possibile specificare una delle tre diverse versioni della crittografia Kerberos utilizzando il parametro `spec.nfsMountOptions`:

- `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 dell'identità e della privacy)

Specificare un solo livello Kerberos. Se si specifica più di un livello di crittografia Kerberos nell'elenco dei parametri, viene utilizzata solo la prima opzione.

Passaggi

1. Sul cluster gestito, creare un file di configurazione del backend di storage utilizzando il seguente esempio. Sostituire i valori tra parentesi <> con le informazioni del proprio 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 passo precedente per creare il backend:

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

Se la creazione del backend fallisce, c'è qualcosa di sbagliato nella configurazione del backend. Puoi visualizzare i log per determinarne 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 storage class

È possibile creare una storage class per il provisioning dei volumi con crittografia Kerberos.

Informazioni su questa attività

Quando si crea un oggetto classe di archiviazione, è possibile specificare una delle tre diverse versioni della crittografia Kerberos utilizzando il parametro `mountOptions`:

- `mountOptions: sec=krb5` (autenticazione e crittografia)
- `mountOptions: sec=krb5i` (autenticazione e crittografia con protezione dell'identità)
- `mountOptions: sec=krb5p` (autenticazione e crittografia con protezione dell'identità e della privacy)

Specificare un solo livello Kerberos. Se si specifica più di un livello di crittografia Kerberos nell'elenco dei parametri, viene utilizzata solo la prima opzione. Se il livello di crittografia specificato nella configurazione del backend di storage è diverso dal livello specificato nell'oggetto storage class, l'oggetto storage class ha la precedenza.

Passaggi

1. Crea un oggetto StorageClass Kubernetes, utilizzando 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. Crea la classe di archiviazione:

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. Assicurati che la storage class sia stata creata:

```
kubectl get sc ontap-nas-sc
```

   Dovresti vedere un output simile al seguente:

```
NAME          PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

Provisioning dei volumi

Dopo aver creato un backend di archiviazione e una classe di archiviazione, è possibile eseguire il provisioning di un volume. Per istruzioni, consultare "Effettua il provisioning di un volume".

Configura la crittografia Kerberos in volo con i volumi Azure NetApp Files

È possibile abilitare la crittografia Kerberos sul traffico di archiviazione tra il cluster gestito e un singolo backend di archiviazione Azure NetApp Files o un pool virtuale di backend di archiviazione Azure NetApp Files.

Prima di iniziare

- Assicurarsi di aver abilitato Trident sul cluster gestito Red Hat OpenShift.
- Assicurati di avere accesso all' `tridentctl` utility.
- Assicurati di aver preparato il backend di archiviazione Azure NetApp Files per la crittografia Kerberos, prendendo nota dei requisiti e seguendo le istruzioni in "Documentazione di 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".

Creare un backend di storage

È possibile creare una configurazione del backend di storage di Azure NetApp Files che include la funzionalità di crittografia Kerberos.

Informazioni su questa attività

Quando si crea un file di configurazione del backend di archiviazione che configura la crittografia Kerberos, è possibile definirlo in modo che venga applicato a uno dei due livelli possibili:

- Il **livello di backend dello storage** utilizzando il `spec.kerberos` campo
- Il **livello del pool virtuale** utilizzando il campo `spec.storage.kerberos`

Quando si definisce la configurazione a livello di pool virtuale, il pool viene selezionato utilizzando l'etichetta nella storage class.

A entrambi i livelli, puoi 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 dell'identità e della privacy)

Passaggi

1. Sul cluster gestito, creare un file di configurazione del backend di archiviazione utilizzando uno dei seguenti esempi, a seconda di dove è necessario definire il backend di archiviazione (livello di backend di archiviazione o livello di pool virtuale). Sostituire i valori tra parentesi <> con le informazioni del proprio ambiente:

Esempio a livello di backend di storage

```
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 di 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 passo precedente per creare il backend:

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

Se la creazione del backend fallisce, c'è qualcosa di sbagliato nella configurazione del backend. Puoi visualizzare i log per determinarne 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 storage class

È possibile creare una storage class per il provisioning dei volumi con crittografia Kerberos.

Passaggi

1. Crea un oggetto StorageClass Kubernetes, utilizzando 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. Crea la classe di archiviazione:

```
kubectl create -f sample-input/storage-class-sc-nfs.yaml
```

3. Assicurati che la storage class sia stata creata:

```
kubectl get sc -sc-nfs
```

Dovresti vedere un output simile al seguente:

```
NAME          PROVISIONER             AGE
sc-nfs        csi.trident.netapp.io   15h
```

Provisioning dei volumi

Dopo aver creato un backend di archiviazione e una classe di archiviazione, è possibile eseguire il provisioning di un volume. Per istruzioni, consultare "Effettua il provisioning di un volume".

Proteggi le applicazioni con Trident Protect

Scopri Trident Protect

NetApp Trident Protect offre avanzate capacità di gestione dei dati delle applicazioni che migliorano la funzionalità e la disponibilità delle applicazioni Kubernetes stateful supportate dai sistemi di storage NetApp ONTAP e dal NetApp Trident CSI storage provisioner. Trident Protect semplifica la gestione, la protezione e la movimentazione dei carichi di lavoro containerizzati tra cloud pubblici e ambienti on-premises. Offre inoltre funzionalità di automazione tramite la sua API e CLI.

È possibile proteggere le applicazioni con Trident Protect creando risorse personalizzate (CR) o utilizzando la Trident Protect CLI.

E ora?

Puoi conoscere i requisiti di Trident Protect prima di installarlo:

- "Requisiti di Trident Protect"

Installa Trident Protect

Requisiti di Trident Protect

Per iniziare, verifica la disponibilità 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 di 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)
- Microsoft Azure Kubernetes Service (AKS)
- Red Hat OpenShift
- SUSE Harvester 1.7.0 (ONTAP iSCSI)
- SUSE Rancher
- VMware Tanzu Portfolio
- Kubernetes upstream

NOTA

- I backup di 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, fai riferimento a "VolumeSnapshotClass".
- Per installare Trident Protect è necessario Helm 4.x o versioni successive.

Compatibilità del backend di storage Trident Protect

Trident Protect supporta i seguenti storage back-end:

- Amazon FSx for NetApp ONTAP
- Cloud Volumes ONTAP
- Array di storage ONTAP
- Google Cloud NetApp Volumes
- Azure NetApp Files

Assicurati che il tuo storage backend soddisfi i seguenti requisiti:

- Assicurarsi che lo storage NetApp connesso al cluster utilizzi Trident 24.02 o una versione successiva (Trident 24.10 è consigliato).
- Assicurati di disporre di un backend di storage NetApp ONTAP.
- Assicurati di aver configurato un bucket di storage a oggetti per l'archiviazione dei backup.
- Crea tutti gli spazi dei nomi applicativi che intendi utilizzare per le applicazioni o per le operazioni di gestione dei dati applicativi. Trident Protect non crea questi spazi dei nomi per te; se specifichi uno spazio dei nomi inesistente in una risorsa personalizzata, l'operazione non riuscirà.

Requisiti per i volumi nas-economy

Trident Protect supporta le operazioni di backup e ripristino su volumi nas-economy. Snapshot, cloni e SnapMirror e la replica su volumi nas-economy non sono attualmente supportati. È necessario abilitare una directory snapshot per ogni volume nas-economy che si prevede di utilizzare con Trident Protect.

NOTA

Alcune applicazioni non sono compatibili con i volumi che utilizzano una directory snapshot. Per queste applicazioni, è necessario nascondere la directory snapshot eseguendo il seguente comando sul sistema storage ONTAP:

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

È possibile abilitare la directory snapshot eseguendo il seguente comando per ciascun volume nas-economy, sostituendo `<volume-UUID>` con l'UUID del volume che si desidera modificare:

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

NOTA	È possibile abilitare le directory snapshot per impostazione predefinita per i nuovi volumi impostando l'opzione di configurazione del backend Trident `snapshotDir` su `true`. I volumi esistenti non sono interessati.

Protezione dei dati con le VM KubeVirt

Trident Protect offre 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 tra le versioni di Trident Protect, con le release più recenti che offrono una configurazione semplificata tramite i parametri del chart Helm.

NOTA	Durante le operazioni di ripristino, qualsiasi `VirtualMachineSnapshots` creato per una macchina virtuale (VM) non viene ripristinato.

Trident Protect 25.10 e versioni successive

Trident Protect blocca e sblocca automaticamente i file system di KubeVirt durante le operazioni di protezione dei dati per garantire la coerenza. A partire da Trident Protect 25.10, puoi disabilitare questo comportamento utilizzando il parametro `vm.freeze` durante l'installazione del chart Helm. Il parametro è abilitato per impostazione predefinita.

```
helm install ... --set vm.freeze=false ...
```

Trident Protect 24.10.1 a 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, puoi disabilitare 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 delle VM KubeVirt utilizzando Trident Protect 24.10, è necessario abilitare manualmente la funzionalità di freeze/unfreeze 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
```

Requisiti per la replicazione SnapMirror

NetApp SnapMirror la replica 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 for NetApp ONTAP

Requisiti del cluster ONTAP per la replica SnapMirror

Assicurarsi che il cluster ONTAP soddisfi i seguenti requisiti se si prevede di utilizzare la replica SnapMirror:

- **NetApp Trident**: NetApp Trident deve essere presente sia sui cluster Kubernetes di origine che su quelli di destinazione che utilizzano ONTAP come backend. Trident Protect supporta la replica con la tecnologia NetApp SnapMirror utilizzando classi di storage supportate dai seguenti driver:
  - `ontap-nas`: NFS
  - `ontap-san`: iSCSI
  - `ontap-san`: FC
  - `ontap-san`: NVMe/TCP (richiede la versione minima di ONTAP 9.15.1)
- **Licenze**: Le licenze asincrone ONTAP SnapMirror che utilizzano il bundle Data Protection devono essere abilitate sia sul cluster ONTAP di origine che su quello di destinazione. Consulta "Panoramica delle licenze SnapMirror in ONTAP" per ulteriori informazioni.

  A partire da ONTAP 9.10.1, tutte le licenze vengono fornite come NetApp license file (NLF), ovvero un singolo file che abilita più funzionalità. Consulta "Licenze incluse con ONTAP One" per maggiori informazioni.

  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 backend di storage:

- **Cluster e SVM**: I backend di storage ONTAP devono essere sottoposti a peering. Consulta "Panoramica del peering di cluster e SVM" per ulteriori informazioni.

NOTA	Assicurarsi che i nomi 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 storage ONTAP in Trident Protect per creare una relazione di replica.

Configurazione di Trident / ONTAP per la replica SnapMirror

Trident Protect richiede che tu configuri almeno un backend di storage che supporti la replica sia per il cluster di origine che per il cluster di destinazione. Se il cluster di origine e il cluster di destinazione sono gli stessi, l'applicazione di destinazione dovrebbe utilizzare un backend di storage diverso da quello dell'applicazione di origine per la migliore resilienza.

Requisiti del cluster Kubernetes per la replica SnapMirror

Assicurati che i tuoi cluster Kubernetes soddisfino i seguenti requisiti:

- **AppVault accessibilità**: sia il cluster di origine che quello di destinazione devono avere accesso alla rete per leggere dalla e scrivere sulla AppVault per la replica degli oggetti dell'applicazione.
- **Connettività di rete**: configura le regole del firewall, le autorizzazioni dei bucket e le liste consentite di IP per consentire la comunicazione tra entrambi i cluster e il AppVault attraverso le WAN.

NOTA	Molti ambienti aziendali implementano rigide policy firewall sulle connessioni WAN. Verificate questi requisiti di rete con il vostro team infrastrutturale prima di configurare la replica.

Installa e configura Trident Protect

Se l'ambiente soddisfa i requisiti per Trident Protect, è possibile seguire questi passaggi per installare Trident Protect sul cluster. È possibile ottenere Trident Protect da NetApp, oppure installarlo dal proprio registro privato. L'installazione da un registro privato è utile se il cluster non può accedere a Internet.

Installa Trident Protect

Installa Trident Protect da NetApp

Passaggi

1. Aggiungi il repository Trident Helm:

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

2. Utilizza Helm per installare Trident Protect. Sostituisci `<name-of-cluster>` con un nome di 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 100.2602.0 --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
100.2602.0 --create-namespace --namespace trident-protect
```

   La registrazione del debug aiuta il supporto NetApp a risolvere i problemi senza richiedere modifiche al livello di registro o la riproduzione del problema.

Installa Trident Protect da un registro privato

Puoi installare Trident Protect da un registro immagini privato se il tuo cluster Kubernetes non è in grado di accedere a Internet. In questi esempi, sostituisci i valori tra parentesi con le informazioni del tuo ambiente:

Passaggi

1. Scarica le seguenti immagini sulla tua macchina locale, aggiorna i tag e poi caricale nel tuo registro privato:

```
docker.io/netapp/controller:26.02.0
docker.io/netapp/restic:26.02.0
docker.io/netapp/kopia:26.02.0
docker.io/netapp/kopiablockrestore:26.02.0
docker.io/netapp/trident-autosupport:26.02.0
docker.io/netapp/exechook:26.02.0
docker.io/netapp/resourcebackup:26.02.0
docker.io/netapp/resourcerestore:26.02.0
docker.io/netapp/resourcedelete:26.02.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

Ad esempio:

```
docker pull docker.io/netapp/controller:26.02.0
```

```
docker tag docker.io/netapp/controller:26.02.0 <private-registry-url>/controller:26.02.0
```

```
docker push <private-registry-url>/controller:26.02.0
```

NOTA Per ottenere il grafico Helm, scarica innanzitutto il grafico Helm su una macchina con accesso a Internet utilizzando `helm pull trident-protect --version 100.2602.0 --repo https://netapp.github.io/trident-protect-helm-chart`, quindi copia il file risultante `trident-protect-100.2602.0.tgz` nel tuo ambiente offline e installalo utilizzando `helm install trident-protect ./trident-protect-100.2602.0.tgz` invece del riferimento al repository nel passaggio finale.

2. Crea lo spazio dei nomi di sistema Trident Protect:

```
kubectl create ns trident-protect
```

3. Accedi al registro:

```
helm registry login <private-registry-url> -u <account-id> -p <api-token>
```

4. Crea un pull secret da utilizzare per l'autenticazione del registro privato:

```
kubectl create secret docker-registry regcred --docker
-username=<registry-username> --docker-password=<api-token> -n
trident-protect --docker-server=<private-registry-url>
```

5. Aggiungi 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`. Assicurati che contenga le seguenti impostazioni di Trident Protect:

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

NOTA I `imageRegistry` e `imagePullSecrets` valori si applicano a tutte le immagini dei componenti, inclusi `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 full path nel campo del registro. Questo garantirà che tutte le immagini vengano estratte da `<private-registry-url>/<image-name>:<tag>`.

7. Utilizza Helm per installare Trident Protect. Sostituisci `<name_of_cluster>` con un nome di 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 100.2602.0 --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
100.2602.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

La registrazione del debug aiuta il supporto NetApp a risolvere i problemi senza richiedere modifiche al livello di registro o la riproduzione del problema.

NOTA	Per ulteriori opzioni di configurazione del grafico Helm, incluse le impostazioni di AutoSupport e il filtraggio dello spazio dei nomi, fare riferimento a "Personalizza l'installazione di Trident Protect".

Installa il plugin Trident Protect CLI

È possibile utilizzare il plugin della riga di comando Trident Protect, che è un'estensione dell'utilità Trident `tridentctl`, per creare e interagire con le risorse personalizzate (CR) di Trident Protect.

Installa il plugin Trident Protect CLI

Prima di utilizzare l'utilità della riga di comando, è necessario installarla sul computer utilizzato per accedere al cluster. Seguire questi passaggi, a seconda che il computer utilizzi una CPU x64 o ARM.

Scarica il plugin per CPU Linux AMD64

Passaggi

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-linux-amd64
```

Scarica il plugin per CPU Linux ARM64

Passaggi

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-linux-arm64
```

Scarica il plugin per CPU Mac AMD64

Passaggi

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-macos-amd64
```

Scarica il plugin per CPU Mac ARM64

Passaggi

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-macos-arm64
```

1. Abilita i permessi di esecuzione per il binario del plugin:

```
chmod +x tridentctl-protect
```

2. Copia il file binario del plugin in una posizione definita nella variabile PATH. Ad esempio, `/usr/bin` o `/usr/local/bin` (potresti aver bisogno di privilegi elevati):

```
cp ./tridentctl-protect /usr/local/bin/
```

3. Facoltativamente, puoi copiare il file binario del plugin in una posizione nella tua home directory. In questo caso, è consigliato assicurarsi che la posizione faccia parte della variabile PATH:

```
cp ./tridentctl-protect ~/bin/
```

NOTA Copiando il plugin in una posizione nella variabile PATH, puoi utilizzare il plugin digitando `tridentctl-protect` o `tridentctl protect` da qualsiasi posizione.

Visualizza la guida del plugin Trident CLI

È possibile utilizzare le funzionalità di aiuto integrate nel plugin per ottenere assistenza dettagliata sulle capacità del plugin:

Passaggi

1. Utilizzare la funzione di aiuto per visualizzare le istruzioni sull'utilizzo:

```
tridentctl-protect help
```

Abilita il completamento automatico dei comandi

Dopo aver installato il plugin Trident Protect CLI, puoi abilitare il completamento automatico per determinati comandi.

Abilita il completamento automatico per la shell Bash

Passaggi

1. Crea lo script di completamento:

```
tridentctl-protect completion bash > tridentctl-completion.bash
```

2. Crea una nuova directory nella tua home directory per contenere lo script:

```
mkdir -p ~/.bash/completions
```

3. Spostare lo script scaricato nella directory `~/.bash/completions`:

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. Aggiungi la seguente riga al `~/.bashrc` file nella tua home directory:

```
source ~/.bash/completions/tridentctl-completion.bash
```

Abilita il completamento automatico per la Z shell

Passaggi

1. Crea lo script di completamento:

```
tridentctl-protect completion zsh > tridentctl-completion.zsh
```

2. Crea una nuova directory nella tua home directory per contenere lo script:

```
mkdir -p ~/.zsh/completions
```

3. Spostare lo script scaricato nella directory `~/.zsh/completions`:

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. Aggiungi la seguente riga al `~/.zprofile` file nella tua home directory:

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

Risultato

Al prossimo accesso alla shell, puoi utilizzare il completamento automatico dei comandi con il plugin tridentctl-protect.

Personalizza l'installazione di Trident Protect

È possibile personalizzare la configurazione predefinita di Trident Protect per soddisfare i requisiti specifici del tuo ambiente.

Specificare i limiti delle risorse del container Trident Protect

È possibile utilizzare un file di configurazione per specificare i limiti delle risorse per i container Trident Protect dopo aver installato Trident Protect. L'impostazione dei limiti delle risorse consente di controllare quante risorse del cluster vengono consumate dalle operazioni di Trident Protect.

Passaggi

1. Crea un file denominato `resourceLimits.yaml`.
2. Compila il file con le opzioni di limitazione delle risorse per i container Trident Protect in base alle esigenze del tuo ambiente.

   Il seguente file di configurazione mostra le impostazioni disponibili e contiene i valori predefiniti per ciascun limite di risorsa:

```
---
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. Applica i valori dal file `resourceLimits.yaml`:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

Personalizza i vincoli del contesto di sicurezza

È possibile utilizzare un file di configurazione per modificare i vincoli di contesto di sicurezza (OpenShift SCC) per i container Trident Protect dopo aver installato Trident Protect. Questi vincoli definiscono le restrizioni di sicurezza per i pod in un cluster OpenShift.

Passaggi

1. Crea un file denominato `sccconfig.yaml`.
2. Aggiungi l'opzione SCC al file e modifica i parametri secondo le esigenze del tuo ambiente.

   Il seguente esempio mostra 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
crea	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 si opera su OpenShift, o se `scc.create` è impostato su `false`, non verrà creata alcuna risorsa SCC.	true
nome	Specifica il nome dell'SCC.	trident-protect-job
priorità	Definisce la priorità dell'SCC. Gli SCC con valori di priorità più alti vengono valutati prima di quelli con valori più bassi.	1

3. Applica i valori dal file `sccconfig.yaml`:

```
helm upgrade trident-protect -n trident-protect netapp-trident-protect/trident-protect -f sccconfig.yaml --reuse-values
```

Questo sostituirà i valori predefiniti con quelli specificati nel `sccconfig.yaml` file.

Configura impostazioni aggiuntive dell'helm chart Trident Protect

È possibile personalizzare le impostazioni di AutoSupport e il filtraggio dello spazio dei nomi per soddisfare i requisiti specifici. La tabella seguente descrive i parametri di configurazione disponibili:

Parametro	Tipo	Descrizione
AutoSupport.proxy	stringa	Configura un URL proxy per le connessioni NetApp AutoSupport. Usa questa opzione per instradare i caricamenti dei pacchetti di supporto attraverso un proxy server. Esempio: `http://my.proxy.url`.
AutoSupport.insecure	booleano	Ignora 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.enabled	booleano	Abilita o disabilita i caricamenti giornalieri dei bundle AutoSupport di Trident Protect. Quando impostato su `false`, i caricamenti giornalieri programmati sono disabilitati, ma è comunque possibile generare manualmente i bundle di supporto. (predefinito: `true`)
restoreSkipNamespaceAnnotations	stringa	Elenco separato da virgole di annotazioni dei namespace da escludere dalle operazioni di backup e ripristino. Consente di filtrare i namespace in base alle annotazioni.
restoreSkipNamespaceLabels	stringa	Elenco separato da virgole di etichette di namespace da escludere dalle operazioni di backup e ripristino. Consente di filtrare i namespace in base alle etichette.

È possibile configurare queste opzioni utilizzando un file di configurazione YAML o i flag della riga di comando:

Usa il file YAML

Passaggi

1. Creare un file di configurazione e chiamarlo `values.yaml`.
2. Nel file che hai 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"
```

3. Dopo aver popolato il file `values.yaml` con i valori corretti, applica il file di configurazione:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

Utilizzare il flag CLI

Passaggi

1. Utilizzare il seguente comando con il flag `--set` per specificare i singoli parametri:

```
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}" \
  --reuse-values
```

Limitare i pod Trident Protect a nodi specifici

È possibile utilizzare il vincolo di selezione dei nodi Kubernetes nodeSelector per controllare quali dei propri 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.

Passaggi

1. Crea un file denominato `nodeSelectorConfig.yaml`.
2. Aggiungere l'opzione nodeSelector al file e modificare il file per aggiungere o cambiare le etichette dei nodi da limitare in base alle esigenze dell'ambiente. Ad esempio, il file seguente contiene la restrizione

predefinita per il sistema operativo, ma anche una regione e un nome di app specifici:

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. Applica i valori dal file `nodeSelectorConfig.yaml`:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

Questo sostituisce le restrizioni predefinite con quelle specificate nel `nodeSelectorConfig.yaml` file.

Gestisci Trident Protect

Gestisci l'autorizzazione e il controllo degli accessi di Trident Protect

Trident Protect utilizza il modello Kubernetes di controllo degli accessi in base al ruolo (RBAC). Per impostazione predefinita, Trident Protect fornisce un singolo namespace di sistema e il relativo account di servizio predefinito. Se hai un'organizzazione con molti utenti o esigenze di sicurezza specifiche, puoi utilizzare le funzionalità RBAC di Trident Protect per ottenere un controllo più granulare sull'accesso a risorse e namespace.

L'amministratore del cluster ha sempre accesso alle risorse nello spazio dei nomi predefinito `trident-protect` e può accedere anche alle risorse in tutti gli altri spazi dei nomi. Per controllare l'accesso alle risorse e alle 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 CR di gestione dei dati applicativi nello spazio dei nomi predefinito `trident-protect`. È necessario creare CR di gestione dei dati applicativi in uno spazio dei nomi dell'applicazione (come best practice, creare CR di gestione dei dati applicativi nello stesso spazio dei nomi dell'applicazione associata).

NOTA

Solo gli amministratori devono avere accesso agli oggetti risorsa personalizzati privilegiati di Trident Protect, che includono:

- **AppVault**: Richiede i dati delle credenziali del bucket
- **AutoSupportBundle**: Raccoglie metriche, log e altri dati sensibili di Trident Protect
- **AutoSupportBundleSchedule**: Gestisce le pianificazioni di raccolta dei log

Come best practice, utilizzare RBAC per limitare l'accesso agli oggetti privilegiati agli amministratori.

Per ulteriori informazioni su come RBAC regola l'accesso alle risorse e ai namespace, consultare la "Documentazione Kubernetes RBAC".

Per informazioni sugli account di servizio, consultare il "Documentazione dell'account di servizio Kubernetes".

Esempio: Gestisci l'accesso per due gruppi di utenti

Ad esempio, un'organizzazione ha un amministratore del cluster, un gruppo di utenti di ingegneria e un gruppo di utenti di marketing. L'amministratore del cluster completerà le seguenti operazioni per creare un ambiente in cui il gruppo di ingegneria e il gruppo di marketing abbiano accesso solo alle risorse assegnate ai rispettivi spazi dei nomi.

Passaggio 1: crea uno spazio dei nomi per contenere le risorse per ciascun gruppo

La creazione di un namespace consente di separare logicamente le risorse e di controllare meglio chi ha accesso a tali risorse.

Passaggi

1. Crea uno spazio dei nomi per il gruppo di ingegneria:

```
kubectl create ns engineering-ns
```

2. Crea uno spazio dei nomi per il gruppo marketing:

```
kubectl create ns marketing-ns
```

Passaggio 2: crea nuovi account di servizio per interagire con le risorse in ogni namespace

Ogni nuovo namespace che crei è dotato di un account di servizio predefinito, ma dovresti creare un account di servizio per ogni gruppo di utenti così da poter suddividere ulteriormente i privilegi tra i gruppi in futuro, se necessario.

Passaggi

1. Crea un account di servizio per il gruppo di ingegneria:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. Crea un account di servizio per il gruppo marketing:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

Passaggio 3: crea un segreto per ogni nuovo account di servizio

Un secret dell'account di servizio viene utilizzato per autenticarsi con l'account di servizio e può essere facilmente eliminato e ricreato se compromesso.

Passaggi

1. Crea un segreto per l'account del servizio di ingegneria:

```
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. Crea un segreto per l'account del servizio 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: crea un oggetto RoleBinding per associare l'oggetto ClusterRole a ciascun nuovo account di servizio

Un oggetto ClusterRole predefinito viene creato quando si installa Trident Protect. È possibile associare questo ClusterRole all'account di servizio creando e applicando un oggetto RoleBinding.

Passaggi

1. Associa il ClusterRole all'account del servizio di ingegneria:

```
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. Associa il ClusterRole all'account del servizio 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: testare le autorizzazioni

Verificare che le autorizzazioni siano corrette.

Passaggi

1. Confermare che gli utenti di ingegneria possano accedere alle risorse di ingegneria:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n engineering-ns
```

2. Confermare che gli utenti di ingegneria 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: Concedi l'accesso agli oggetti AppVault

Per eseguire attività di gestione dei dati quali backup e snapshot, l'amministratore del cluster deve concedere l'accesso agli oggetti AppVault ai singoli utenti.

Passaggi

1. Crea e applica un file YAML di combinazione tra AppVault e secret che garantisce a un utente l'accesso a un AppVault. Ad esempio, la seguente CR concede l'accesso a un 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. Crea e applica un Role 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. Crea e applica una RoleBinding CR per associare i permessi 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 sugli oggetti AppVault per tutti gli spazi dei nomi:

   ```
   kubectl get appvaults -n trident-protect
   --as=system:serviceaccount:engineering-ns:eng-user
   ```

   Dovresti vedere un output simile al seguente:

```
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 riesce a ottenere le informazioni AppVault a cui ora ha il permesso di 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
```

Dovresti vedere un output simile al seguente:

```
yes
```

Risultato

Gli utenti a cui hai concesso le autorizzazioni AppVault devono essere in grado di utilizzare oggetti AppVault autorizzati per le operazioni di gestione dei dati dell'applicazione e non devono essere in grado di accedere a nessuna risorsa al di fuori degli spazi dei nomi assegnati o di creare nuove risorse a cui non hanno accesso.

Monitorare le risorse di Trident Protect

È possibile utilizzare gli strumenti open source kube-state-metrics, Prometheus e Alertmanager per monitorare lo stato di salute delle risorse protette da Trident Protect.

Il servizio kube-state-metrics genera metriche dalla comunicazione API di Kubernetes. Utilizzarlo con Trident Protect espone 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 offrono un modo per monitorare l'integrità e lo stato delle risorse che gestisci con Trident Protect.

Alertmanager è un servizio che acquisisce gli avvisi inviati da strumenti come Prometheus e li instrada verso destinazioni che configuri.

NOTA

Le configurazioni e le istruzioni incluse in questi passaggi sono solo esempi; è necessario personalizzarle per adattarle al proprio ambiente. Fare riferimento alla seguente documentazione ufficiale per istruzioni e supporto specifici:

- "documentazione kube-state-metrics"
- "Documentazione di Prometheus"
- "Documentazione di Alertmanager"

Passaggio 1: installa gli strumenti di monitoraggio

Per abilitare il monitoraggio delle risorse in Trident Protect, è necessario installare e configurare kube-state-metrics, Prometheus e Alertmanager.

Installa kube-state-metrics

Puoi installare kube-state-metrics usando Helm.

Passaggi

1. Aggiungi il grafico Helm kube-state-metrics. Ad esempio:

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update
```

2. Applica il CRD ServiceMonitor di Prometheus al cluster:

```
kubectl apply -f https://raw.githubusercontent.com/prometheus-
operator/prometheus-operator/main/example/prometheus-operator-
crd/monitoring.coreos.com_servicemonitors.yaml
```

3. Crea un file di configurazione per il grafico Helm (ad esempio, `metrics-config.yaml`). Puoi personalizzare la seguente configurazione di esempio in base al tuo ambiente:

metrics-config.yaml: configurazione Helm chart 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. Installa kube-state-metrics distribuendo il grafico Helm. Ad esempio:

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. Configura kube-state-metrics per generare metriche per le risorse personalizzate utilizzate da Trident Protect seguendo le istruzioni nel "documentazione delle risorse personalizzate kube-state-metrics".

Installa Prometheus

È possibile installare Prometheus seguendo le istruzioni nel "Documentazione di Prometheus".

Installa Alertmanager

È possibile installare Alertmanager seguendo le istruzioni nel "Documentazione di Alertmanager".

Passaggio 2: configura gli strumenti di monitoraggio affinché funzionino insieme

Dopo aver installato gli strumenti di monitoraggio, è necessario configurarli affinché funzionino insieme.

Passaggi

1. Integra kube-state-metrics con Prometheus. Modifica il file di configurazione Prometheus (`prometheus.yaml` e aggiungi le informazioni sul 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. Configura Prometheus per indirizzare gli avvisi ad Alertmanager. Modifica il file di configurazione di Prometheus (`prometheus.yaml` e aggiungi la seguente sezione:

prometheus.yaml: Invia avvisi ad Alertmanager

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager.trident-protect.svc:9093
```

Risultato

Prometheus ora può raccogliere metriche da kube-state-metrics e inviare avvisi ad Alertmanager. Ora sei pronto per configurare quali condizioni attivano un avviso e dove devono essere inviati gli avvisi.

Passaggio 3: Configura gli avvisi e le destinazioni degli avvisi

Dopo aver configurato gli strumenti affinché funzionino insieme, è necessario configurare quale tipo di informazioni attiva gli avvisi e dove devono essere inviati.

Esempio di avviso: errore di backup

L'esempio seguente definisce 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 al proprio ambiente e includere questo frammento YAML nel file di configurazione `prometheus.yaml`:

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."
```

Configura Alertmanager per inviare avvisi ad altri canali

È possibile configurare Alertmanager per inviare notifiche ad altri canali, come e-mail, PagerDuty, Microsoft Teams o altri servizi di notifica, specificando la rispettiva configurazione nel `alertmanager.yaml` file.

L'esempio seguente configura Alertmanager per inviare notifiche a un canale Slack. Per personalizzare questo esempio in base al tuo ambiente, sostituisci il valore della `api_url` chiave con l'URL del webhook Slack utilizzato nel tuo ambiente:

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
```

Generare un bundle di supporto Trident Protect

Trident Protect consente agli amministratori di generare bundle che includono informazioni utili a NetApp Support, tra cui log, metriche e informazioni sulla topologia dei cluster e delle app gestite. Se si è connessi a Internet, è possibile caricare i bundle di supporto sul NetApp Support Site (NSS) utilizzando un file custom resource (CR).

Crea un bundle di supporto utilizzando un CR

Passaggi

1. Creare il file custom resource (CR) e assegnargli un nome (ad esempio `trident-protect-support-bundle.yaml`).
2. Configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.triggerType**: (*Required*) Determina se il bundle di supporto viene generato immediatamente o programmato. La generazione del bundle programmato avviene alle 12AM UTC. Valori possibili:
     - In programma
     - Manuale
   - **spec.uploadEnabled**: (*Opzionale*) Controlla se il bundle di supporto deve essere caricato sul NetApp Support Site dopo essere stato generato. Se non specificato, il valore predefinito è `false`. Valori possibili:
     - true
     - false (predefinito)
   - **spec.dataWindowStart**: (*Opzionale*) Una stringa di data nel formato RFC 3339 che specifica la data e l'ora in cui deve iniziare la finestra dei dati inclusi nel support bundle. Se non specificato, il valore predefinito è 24 ore fa. La data più remota della finestra che si può 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, applica la CR:

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-protect
```

Crea un bundle di supporto utilizzando la CLI

Passaggi

1. Crea il bundle di supporto, sostituendo i valori tra parentesi con le informazioni del tuo ambiente. Il `trigger-type` determina se il bundle viene creato immediatamente o se l'orario di creazione è

dettato 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
```

Monitora e recupera il pacchetto di supporto

Dopo aver creato un support bundle con uno dei due metodi, puoi monitorare l'avanzamento della generazione e recuperarlo sul tuo sistema locale.

Passaggi

1. Attendere che `status.generationState` raggiunga lo stato `Completed`. È possibile monitorare l'avanzamento della generazione con il seguente comando:

   ```
   kubectl get autosupportbundle trident-protect-support-bundle -n trident-protect
   ```

2. Recupera il bundle di supporto nel sistema locale. Ottieni il comando di copia dal bundle completato AutoSupport:

   ```
   kubectl describe autosupportbundle trident-protect-support-bundle -n trident-protect
   ```

   Individua il comando `kubectl cp` dall'output ed eseguilo, sostituendo l'argomento destination con la tua directory locale preferita.

Aggiorna Trident Protect

Puoi aggiornare Trident Protect all'ultima versione per sfruttare le nuove funzionalità o correzioni di bug.

NOTA

- Quando si esegue l'aggiornamento dalla versione 24.10, gli snapshot in esecuzione durante l'aggiornamento potrebbero non riuscire. Questo errore non impedisce la creazione di snapshot futuri, sia manuali che pianificati. Se uno snapshot non riesce durante l'aggiornamento, è possibile crearne uno nuovo manualmente per garantire che l'applicazione sia protetta.

  Per evitare potenziali errori, è possibile disabilitare tutte le pianificazioni degli Snapshot prima dell'aggiornamento e riabilitarle in seguito. Tuttavia, questo comporta la mancata acquisizione di eventuali Snapshot pianificati durante il periodo di aggiornamento.

- Per le installazioni con registro privato, assicurati che il chart e le immagini Helm richiesti per la versione di destinazione siano disponibili nel tuo registro privato e verifica che i valori Helm personalizzati siano compatibili con la nuova versione del chart. Per ulteriori informazioni, fai riferimento a "Installa Trident Protect da un registro privato".

Passaggio 1: seleziona 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 desideri eseguire l'aggiornamento alla versione 26.02:
  a. Primo upgrade dal 24.10 al 25.02.
  b. Quindi esegui l'aggiornamento da 25.02 a 26.02.

Passaggio 2: Aggiornare Trident Protect

Per aggiornare Trident Protect, eseguire i seguenti passaggi.

Passaggi

1. Aggiorna il repository Helm di Trident:

```
helm repo update
```

2. Aggiorna i CRD di 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 nel chart Helm di Trident Protect.

   a. Esegui 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 il `trident-protect-crds` chart:

NOTA Non disinstallare il `trident-protect-crds` chart tramite Helm, poiché ciò potrebbe rimuovere i tuoi CRD e qualsiasi dato correlato.

```
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 100.2602.0 --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 di debug è consigliata per la risoluzione dei problemi, in quanto aiuta NetApp a diagnosticare i problemi senza richiedere modifiche al livello di registrazione o la riproduzione dei problemi.

Gestire e proteggere le applicazioni

Utilizzare gli oggetti Trident Protect AppVault per gestire i bucket

La risorsa personalizzata (CR) del bucket per Trident Protect è nota come AppVault. Gli oggetti AppVault sono la rappresentazione dichiarativa del workflow Kubernetes di un bucket di storage. Un CR AppVault contiene le configurazioni necessarie affinché un bucket possa essere utilizzato nelle operazioni di protezione, come backup, snapshot, operazioni di ripristino e replica SnapMirror. Solo gli amministratori possono creare AppVaults.

È necessario creare una AppVault CR manualmente o dalla riga di comando quando si eseguono operazioni di protezione dei dati su un'applicazione. La AppVault CR è specifica per il proprio ambiente e puoi utilizzare gli esempi in questa pagina come guida quando crei AppVault CR.

NOTA Assicurarsi che la AppVault CR sia presente sul cluster in cui è installato Trident Protect. Se la AppVault CR non esiste o non è possibile accedervi, la riga di comando mostra un errore.

Configura l’autenticazione e le password di AppVault

Prima di creare una AppVault CR, assicurati che la AppVault e il data mover che scegli possano autenticarsi con il provider e con tutte le risorse correlate.

Password del repository del data mover

Quando si creano oggetti AppVault utilizzando le CR o il plugin Trident Protect CLI, è possibile specificare un segreto Kubernetes con password personalizzate per la crittografia Restic e Kopia. Se non si specifica un segreto, Trident Protect utilizza una password predefinita.

- Quando si creano manualmente i AppVault CR, utilizzare il campo **spec.dataMoverPasswordSecretRef** per specificare il segreto.
- Quando si creano oggetti AppVault utilizzando la Trident Protect CLI, utilizzare l' `--data-mover-password-secret-ref`argomento per specificare il segreto.

Crea un secret per la password del repository del data mover

Utilizzare i seguenti esempi per creare la password segreta. Quando si creano oggetti AppVault, è possibile istruire Trident Protect a utilizzare questa password segreta per l’autenticazione con il repository del data mover.

NOTA

- A seconda del data mover utilizzato, è sufficiente includere la password corrispondente per quel data mover. Ad esempio, se si utilizza Restic e non si prevede di utilizzare Kopia in futuro, è possibile includere solo la password Restic quando si crea il segreto.
- Conserva la password in un luogo sicuro. Ti servirà per ripristinare i dati sullo stesso cluster o su uno diverso. Se il cluster o il `trident-protect` namespace viene eliminato, non potrai 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
```

Usa 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 di storage compatibile S3

Quando si accede a uno storage 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 questa policy all'utente che gestisce le policy 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 nel "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 di Kopia. Questa funzionalità consente l'accesso sicuro ai bucket S3 senza memorizzare le credenziali AWS nei segreti di Kubernetes.

Requisiti per EKS Pod Identity con Trident Protect

Prima di utilizzare EKS Pod Identity con Trident Protect, assicurarsi di quanto segue:

- Il tuo cluster EKS ha abilitato Pod Identity.
- Hai creato un ruolo IAM con le necessarie autorizzazioni per il bucket S3. Per ulteriori informazioni, consultare "Autorizzazioni IAM di storage compatibile S3".
- Il ruolo IAM è associato ai seguenti account del 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, consultare la "Documentazione AWS EKS Pod Identity".

AppVault Configuration Quando si usa EKS Pod Identity, configura il tuo AppVault CR con il flag `useIAM: true` 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
```

AppVault esempi di generazione chiave per cloud provider

Quando si definisce un AppVault CR, è necessario includere le credenziali per accedere alle risorse ospitate dal provider, a meno che non si utilizzi IAM authentication. Le modalità di generazione delle chiavi per le credenziali variano a seconda del provider. Di seguito sono riportati esempi di generazione di chiavi da riga di comando per diversi provider. È possibile utilizzare i seguenti esempi per creare chiavi per le credenziali di ciascun cloud provider.

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
```

S3 generico

```
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 riportati esempi di definizioni AppVault per ciascun provider.

AppVault esempi di CR

È possibile utilizzare i seguenti esempi di CR per creare oggetti AppVault per ciascun cloud provider.

NOTA

- È possibile specificare facoltativamente un segreto Kubernetes che contiene password personalizzate per la crittografia dei repository Restic e Kopia. Consulta Password del repository del data mover per maggiori informazioni.
- Per gli oggetti Amazon S3 (AWS) AppVault, puoi specificare facoltativamente un sessionToken, che è utile se utilizzi il single sign-on (SSO) per l'autenticazione. Questo token viene creato quando generi le chiavi per il provider in AppVault esempi di generazione chiave per cloud provider.
- Per gli oggetti S3 AppVault, puoi facoltativamente specificare un URL proxy di egress per il traffico S3 outbound utilizzando la chiave `spec.providerConfig.S3.proxyURL`.

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, puoi rimuovere la sezione `providerCredentials` e aggiungere `useIAM: true` sotto la configurazione `s3`.

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
```

S3 generico

```
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 di Trident Protect

È possibile utilizzare i seguenti esempi di comando CLI per creare AppVault CR per ciascun provider.

NOTA

- È possibile specificare facoltativamente un segreto Kubernetes che contiene password personalizzate per la crittografia dei repository Restic e Kopia. Consulta Password del repository del data mover per maggiori informazioni.
- Per gli oggetti S3 AppVault, puoi facoltativamente specificare un URL proxy di egress per il traffico S3 outbound utilizzando l'argomento `--proxy-url <ip_address:port>`.

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
```

S3 generico

```
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
```

Opzioni di configurazione `providerConfig.s3` supportate

Consultare la tabella seguente per le opzioni di configurazione del provider S3:

Parametro	Descrizione	Predefinito	Esempio
`providerConfig.s3.skipCertValidation`	Disattiva la verifica dei certificati SSL/TLS.	falso	"true", "false"
`providerConfig.s3.secure`	Abilita la comunicazione sicura HTTPS con l'endpoint S3.	true	"true", "false"
`providerConfig.s3.proxyURL`	Specificare l'URL del proxy server utilizzato per connettersi a S3.	Nessuno	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	Fornisci un certificato root CA personalizzato per la verifica SSL/TLS.	Nessuno	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	Abilita l'autenticazione IAM per accedere ai bucket S3. Applicabile per EKS Pod Identity.	falso	vero, falso

Visualizza informazioni AppVault

È possibile utilizzare il plugin Trident Protect CLI per visualizzare le informazioni sugli oggetti AppVault che hai creato sul cluster.

Passaggi

1. Visualizza il contenuto di un AppVault object:

```
tridentctl-protect get appvaultcontent gcp-vault \
--show-resources all \
-n trident-protect
```

Esempio output:

```
+-------------+-------+----------+-----------------------------
+---------------------------+
|   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. Opzionalmente, per vedere il AppVaultPath per ogni risorsa, usa il flag `--show-paths`.

   Il nome del cluster nella prima colonna della tabella è disponibile solo se un nome del cluster è stato specificato nell'installazione di Trident Protect helm. Ad esempio: `--set clusterName=production1`.

Rimuovi un AppVault

È possibile rimuovere un AppVault oggetto in qualsiasi momento.

NOTA	Non rimuovere la `finalizers` chiave nel AppVault CR prima di eliminare l'oggetto AppVault. In caso contrario, potrebbero rimanere dati residui nel bucket AppVault e risorse orfane nel cluster.

Prima di iniziare

Assicurati di aver eliminato tutti gli snapshot e i backup CR utilizzati da AppVault che desideri eliminare.

Rimuovi un AppVault utilizzando la CLI di Kubernetes

1. Rimuovi l'oggetto AppVault, sostituendo `appvault-name` con il nome dell'oggetto AppVault da rimuovere:

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

Rimuovi un AppVault utilizzando la Trident Protect CLI

1. Rimuovi l'oggetto AppVault, sostituendo `appvault-name` con il nome dell'oggetto AppVault da rimuovere:

```
tridentctl-protect delete appvault <appvault-name> \
-n 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.

Crea una AppVault CR

È necessario creare una AppVault CR che verrà utilizzata durante l'esecuzione delle operazioni di protezione dei dati sull'applicazione e la AppVault CR deve risiedere nel cluster in cui è installato Trident Protect. La AppVault CR è specifica per il tuo ambiente; per esempi di AppVault CR, 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 una CR dell'applicazione o utilizzando la Trident Protect CLI.

Aggiungi un'applicazione utilizzando un CR

Passaggi

1. Crea il file CR dell'applicazione di destinazione:
   a. Creare il file custom resource (CR) e assegnargli un nome (ad esempio `maria-app.yaml`).
   b. Configura i seguenti attributi:
      - **metadata.name**: (*Required*) Il nome della risorsa personalizzata dell'applicazione. Nota il nome che scegli perché altri file CR necessari per le operazioni di protezione fanno riferimento a questo valore.
      - **spec.includedNamespaces**: (*Obbligatorio*) Utilizzare il selettore di namespace e di etichetta per specificare i namespace e le risorse che l'applicazione utilizza. Il namespace dell'applicazione deve essere parte di questo elenco. Il selettore di etichetta è facoltativo e può essere utilizzato per filtrare le risorse all'interno di ciascun namespace specificato.
      - **spec.includedClusterScopedResources**: (*Facoltativo*) 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**: (*Obbligatorio*) Specifica il gruppo API, la versione e il tipo di risorsa con ambito cluster.
        - **labelSelector**: (*Facoltativo*) Filtra le risorse con ambito cluster in base alle loro etichette.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Opzionale*) Questa annotazione è applicabile solo alle applicazioni definite da macchine virtuali, ad esempio in ambienti KubeVirt, in cui si verificano blocchi del filesystem prima degli snapshot. Specifica se questa applicazione può scrivere sul filesystem durante uno snapshot. Se impostato su true, l'applicazione ignora l'impostazione globale e può scrivere sul filesystem durante uno snapshot. Se impostato su false, l'applicazione ignora l'impostazione globale e il filesystem 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 "impostazione globale di congelamento Trident Protect".

Se è necessario applicare questa annotazione dopo che un'applicazione è già stata creata, 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**: (Obbligatorio per il filtraggio) Utilizzare `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 resourceMatcher oggetti. Se si definiscono più elementi in questo array, la corrispondenza avviene tramite un'operazione OR, e i campi all'interno di ciascun elemento (group, kind, version) corrispondono tramite un'operazione AND.
       - **resourceMatchers[].group**: (*Facoltativo*) Gruppo della risorsa da filtrare.
       - **resourceMatchers[].kind**: (*Facoltativo*) Tipo di risorsa da filtrare.

- **resourceMatchers[].version**: (*Facoltativo*) Versione della risorsa da filtrare.
- **resourceMatchers[].names**: (*Facoltativo*) Nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].namespaces**: (*Facoltativo*) Namespace nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].labelSelectors**: (*Facoltativo*) Stringa del selettore di etichetta nel campo metadata.name dei metadati Kubernetes della risorsa come definito in "Documentazione Kubernetes". Ad esempio: `"trident.netapp.io/os=linux"`.

NOTA	Quando sia `resourceFilter` che `labelSelector` vengono utilizzati, `resourceFilter` viene eseguito per primo 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 la CR dell'applicazione adatta al tuo ambiente, applica la CR. Ad esempio:

```
kubectl apply -f maria-app.yaml
```

Passaggi

1. Crea e applica la definizione dell'applicazione utilizzando uno dei seguenti esempi, sostituendo i valori tra parentesi con le informazioni del tuo ambiente. Puoi includere namespace e risorse nella definizione dell'applicazione utilizzando elenchi separati da virgole con gli argomenti mostrati negli esempi.

   Facoltativamente, puoi utilizzare un'annotazione quando crei un'app per specificare se l'applicazione può scrivere sul filesystem durante uno snapshot. Questo è applicabile solo alle applicazioni definite da macchine virtuali, come negli ambienti KubeVirt, dove il filesystem viene bloccato prima degli snapshot. Se imposti l'annotazione su `true`, l'applicazione ignora l'impostazione globale e può scrivere sul filesystem durante uno snapshot. Se la imposti su `false`, l'applicazione ignora l'impostazione globale e il filesystem viene bloccato durante uno snapshot. Se utilizzi l'annotazione ma l'applicazione non ha macchine virtuali nella definizione dell'applicazione, l'annotazione viene ignorata. Se non utilizzi l'annotazione, l'applicazione segue "impostazione globale di congelamento Trident Protect".

   Per specificare l'annotazione quando si utilizza la CLI per creare un'applicazione, è possibile utilizzare il `--annotation` flag.

   - Crea l'applicazione e utilizza l'impostazione globale per il comportamento di blocco del file system:

```
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 l'impostazione dell'applicazione locale per il comportamento di freeze 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">
```

- È possibile utilizzare `--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 nell'esempio seguente:

```
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, 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:

```
tridentctl-protect create app my-pvc-app --namespaces <my-app-
namespace> --resource-filter-include
'[{"Kind":"PersistentVolumeClaim","Version":"v1"},{"Kind":"Persis
tentVolume","Version":"v1"},{"Kind":"VolumeSnapshotClass","Versio
n":"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 utilizzando una policy di protezione automatizzata o su base ad hoc.

NOTA	È possibile configurare Trident Protect per congelare e scongelare i filesystem durante le operazioni di protezione dei dati. "Scopri di più sulla configurazione del congelamento del filesystem con Trident Protect".

Crea un'istantanea su richiesta

È possibile creare una snapshot su richiesta in qualsiasi momento.

NOTA	Le risorse con ambito cluster vengono incluse in un backup, snapshot o clone se sono esplicitamente referenziate nella definizione dell'applicazione o se hanno riferimenti a uno qualsiasi degli spazi dei nomi dell'applicazione.

Crea un'istantanea utilizzando un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-snapshot-cr.yaml`.
2. Nel file che hai creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.applicationRef**: Nome Kubernetes dell'applicazione di cui eseguire lo Snapshot.
   - **spec.appVaultRef**: (*Obbligatorio*) Il nome del AppVault in cui i contenuti dello snapshot (metadati) devono essere archiviati.
   - **spec.reclaimPolicy**: (*Facoltativo*) Definisce cosa succede all'AppArchive di uno snapshot quando il CR dello snapshot viene eliminato. Ciò significa che anche quando impostato su `Retain`, lo snapshot verrà eliminato. Opzioni valide:
     - `Retain` (predefinito)
     - `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, applica la CR:

   ```
   kubectl apply -f trident-protect-snapshot-cr.yaml
   ```

Crea un'istantanea utilizzando la CLI

Passaggi

1. Crea l'istantanea, sostituendo i valori tra parentesi con le informazioni del tuo 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

È possibile eseguire il backup di un'app in qualsiasi momento.

NOTA	Le risorse con ambito cluster vengono incluse in un backup, snapshot o clone se sono esplicitamente referenziate nella definizione dell'applicazione o se hanno riferimenti a uno qualsiasi degli spazi dei nomi dell'applicazione.

Prima di iniziare

Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di backup s3 di lunga durata. Se il token scade durante l'operazione di backup, l'operazione può fallire.

- Fare riferimento a "Documentazione API AWS" per ulteriori informazioni sulla verifica della scadenza del token della sessione corrente.
- Fare riferimento a "Documentazione IAM AWS" per ulteriori informazioni sulle credenziali con le risorse AWS.

Crea un backup utilizzando un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-backup-cr.yaml`.
2. Nel file che hai creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.applicationRef**: (*Obbligatorio*) Il nome Kubernetes dell'applicazione di cui eseguire il backup.
   - **spec.appVaultRef**: (*Required*) Il nome del AppVault in cui devono essere archiviati i contenuti del backup.
   - **spec.dataMover**: (*Opzionale*) Una stringa che indica quale strumento di backup utilizzare per l'operazione di backup. Valori possibili (case sensitive):
     - `Restic`
     - `Kopia` (predefinito)
   - **spec.reclaimPolicy**: (*Opzionale*) Definisce cosa succede a un backup quando viene rilasciato dalla sua richiesta. Valori possibili:
     - `Delete`
     - `Retain` (predefinito)
   - **spec.snapshotRef**: (*Opzionale*): Nome dell'istantanea da usare come origine del backup. Se non fornito, verrà creata un'istantanea temporanea e verrà eseguito il backup.

     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, applica la CR:

```
kubectl apply -f trident-protect-backup-cr.yaml
```

Crea un backup utilizzando la CLI

Passaggi

1. Crea il backup, sostituendo i valori tra parentesi con le informazioni del tuo 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 facoltativamente 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 flag, il backup diventa non incrementale. È best practice eseguire periodicamente un backup completo e poi eseguire backup incrementali tra un backup completo e l'altro per ridurre al minimo il rischio associato ai ripristini.

Annotazioni di backup supportate

La tabella seguente descrive le annotazioni che puoi utilizzare quando crei un backup CR:

Annotazione	Tipo	Descrizione	Valore predefinito
protect.trident.netapp.io/full-backup	stringa	Specifica se un backup deve essere non incrementale. Impostare su `true` per creare un backup non incrementale. È best practice eseguire periodicamente un backup completo e poi eseguire backup incrementali tra un backup completo e l'altro per ridurre al minimo il rischio associato ai ripristini.	"false"
protect.trident.netapp.io/snapshot-completion-timeout	stringa	Il tempo massimo consentito per il completamento dell'operazione di snapshot complessiva.	"60m"
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.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	stringa	Il 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 eventuali nuove PersistentVolumeClaims (PVC) raggiungano la fase `Bound` prima che l'operazione fallisca.	"1200" (20 minuti)

Crea una pianificazione della protezione dei dati

Una policy di protezione protegge un'app creando snapshot, backup o entrambi secondo una pianificazione definita. Puoi scegliere di creare snapshot e backup ogni ora, ogni giorno, ogni settimana e ogni mese, e puoi specificare il numero di copie da conservare. Puoi pianificare un backup completo non incrementale utilizzando l'annotazione full-backup-rule. Per impostazione predefinita, tutti i backup sono incrementali. Eseguire periodicamente un backup completo, insieme a backup incrementali tra un backup completo e l'altro, aiuta a ridurre il rischio associato ai ripristini.

NOTA

- È possibile creare pianificazioni per le istantanee solo impostando `backupRetention` su zero e `snapshotRetention` su un valore maggiore di zero. Impostare `snapshotRetention` su zero significa che qualsiasi backup pianificato creerà comunque istantanee, ma queste sono temporanee e vengono eliminate immediatamente dopo il completamento del backup.
- Le risorse con ambito cluster vengono incluse in un backup, snapshot o clone se sono esplicitamente referenziate nella definizione dell'applicazione o se hanno riferimenti a uno qualsiasi degli spazi dei nomi dell'applicazione.

Crea una pianificazione utilizzando un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-schedule-cr.yaml`.
2. Nel file che hai creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.dataMover**: (*Opzionale*) Una stringa che indica quale strumento di backup utilizzare per l'operazione di backup. Valori possibili (case sensitive):
     - `Restic`
     - `Kopia` (predefinito)
   - **spec.applicationRef**: il nome Kubernetes dell'applicazione di cui eseguire il backup.
   - **spec.appVaultRef**: (*Required*) Il nome del AppVault in cui devono essere archiviati i contenuti del backup.
   - **spec.backupRetention**: (*Required*) Il numero di backup da conservare. Zero indica che non devono essere creati backup (solo istantanee).
   - **backupReclaimPolicy**: (*Opzionale*) Determina cosa succede a un backup se il CR di backup viene eliminato durante il suo periodo di conservazione. Dopo il periodo di conservazione, i backup vengono sempre eliminati. Valori possibili (case sensitive):
     - `Retain` (predefinito)
     - `Delete`
   - **spec.snapshotRetention**: (*Required*) Il numero di istantanee da conservare. Zero indica che non devono essere create istantanee.
   - **snapshotReclaimPolicy**: (*Opzionale*) Determina cosa succede a una snapshot se la CR della snapshot viene eliminata durante il suo periodo di conservazione. Dopo il periodo di conservazione, le snapshot vengono sempre eliminate. Valori possibili (case sensitive):
     - `Retain`
     - `Delete` (predefinito)
   - **spec.granularity**: La frequenza con cui la pianificazione deve essere eseguita. Valori possibili, con i campi associati richiesti:
     - `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 deve essere eseguita la pianificazione. 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 la 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 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, applica la CR:

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

Crea una pianificazione utilizzando la CLI

Passaggi

1. Crea la pianificazione della protezione, sostituendo i valori tra parentesi con le informazioni del tuo ambiente. Ad esempio:

NOTA Puoi usare `tridentctl-protect create schedule --help` per visualizzare informazioni di aiuto dettagliate 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 backup completo**: utilizzare il `--full-backup-rule` flag per pianificare backup completi non incrementali. Questo flag 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` 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`.
- **Pianificazioni solo snapshot**: imposta `--backup-retention 0` e specifica un valore maggiore di zero per `--snapshot-retention`.

Annotazioni di pianificazione supportate

La tabella seguente descrive le annotazioni che è possibile utilizzare quando si crea una schedule CR:

Annotazione	Tipo	Descrizione	Valore predefinito
protect.trident.netapp.io/full-backup-rule	stringa	Specifica la regola per la pianificazione dei backup completi. Puoi impostarla su `Always` per backup completi costanti o personalizzarla in base alle tue esigenze. Ad esempio, se scegli la granularità giornaliera, puoi specificare i giorni della settimana in cui deve essere eseguito il backup completo (ad esempio, `"Monday,Thursday"`). I valori validi per i giorni della settimana sono: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday. Nota 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	Il tempo massimo consentito per il completamento dell'operazione di snapshot complessiva.	"60m"
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.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	stringa	Il 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 eventuali nuove PersistentVolumeClaims (PVC) raggiungano la fase `Bound` prima che l'operazione fallisca.	"1200" (20 minuti)

Elimina una snapshot

Elimina gli snapshot pianificati o on-demand di cui non hai più bisogno.

Passaggi

1. Rimuovi lo snapshot CR associato allo snapshot:

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

Elimina un backup

Elimina i backup pianificati o on-demand di cui non hai più bisogno.

NOTA Assicurarsi che la policy di reclaim sia impostata su `Delete` per rimuovere tutti i dati di backup dallo storage a oggetti. L'impostazione predefinita della policy è `Retain` per evitare la perdita accidentale di dati. Se la policy non viene modificata su `Delete`, i dati di backup rimarranno nello storage a oggetti e richiederanno l'eliminazione manuale.

Passaggi

1. Rimuovi il CR di backup associato al backup:

```
kubectl delete backup <backup_name> -n my-app-namespace
```

Verificare 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.

Passaggi

1. Utilizzare il seguente comando per recuperare lo stato dell'operazione di backup, sostituendo i valori tra parentesi con le informazioni dal proprio ambiente:

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath
='{.status}'
```

Abilita backup e ripristino per le operazioni azure-netapp-files (ANF)

Se hai installato Trident Protect, puoi abilitare la funzionalità di backup e ripristino efficiente in termini di spazio per i backend di storage che utilizzano la storage class 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

Assicurarsi quanto segue:

- Hai installato Trident Protect.
- Hai definito un'applicazione in Trident Protect. Questa applicazione avrà funzionalità di protezione limitate finché non completi questa procedura.
- Hai `azure-netapp-files` selezionato come classe di archiviazione predefinita per il tuo backend di archiviazione.

Espandi per i passaggi di configurazione

1. Eseguire le seguenti operazioni in Trident se il volume ANF è stato creato prima dell'aggiornamento a Trident 24.10:

   a. Abilita la directory snapshot per ogni PV basato su azure-netapp-files e associato all'applicazione:

   ```
   tridentctl update volume <pv name> --snapshot-dir=true -n trident
   ```

   b. Verificare che la directory snapshot sia stata abilitata per ciascun 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 seleziona la normale funzionalità di backup, che consuma temporaneamente spazio nel pool di capacità durante il processo di backup. In questo caso, assicurarsi che sia disponibile spazio sufficiente nel pool di capacità per creare un volume temporaneo delle stesse dimensioni del volume sottoposto a backup.

Risultato

L'applicazione è pronta per backup e ripristino tramite Trident Protect. Ogni PVC è disponibile anche per essere utilizzato da altre applicazioni per backup e ripristino.

Ripristina le 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 quando si ripristina l'applicazione sullo stesso cluster.

NOTA

- Quando si ripristina un'applicazione, tutti gli hook di esecuzione configurati per l'applicazione vengono ripristinati con l'app. Se è presente un execution hook 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 ulteriori informazioni, consultare "Utilizza le impostazioni di ripristino avanzate di Trident Protect".

Ripristina da un backup a un namespace diverso

Quando si ripristina un backup in un namespace diverso utilizzando una BackupRestore CR, Trident Protect ripristina l'applicazione in un nuovo namespace e crea una application CR per l'applicazione ripristinata. Per proteggere l'applicazione ripristinata, crea backup o snapshot on-demand oppure stabilisci una pianificazione di protezione.

NOTA

- Il ripristino di un backup in un namespace diverso con risorse esistenti non modificherà le risorse che condividono i nomi con quelle nel backup. Per ripristinare tutte le risorse nel backup, eliminare e ricreare il namespace di destinazione o ripristinare il backup in un nuovo namespace.
- Quando si utilizza una CR per ripristinare in 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 la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino s3 di lunga durata. Se il token scade durante l'operazione di ripristino, l'operazione può fallire.

- Fare riferimento a "Documentazione API AWS" per ulteriori informazioni sulla verifica della scadenza del token della sessione corrente.
- Fare riferimento a "Documentazione IAM AWS" per ulteriori informazioni sulle credenziali con le risorse AWS.

NOTA

Quando si ripristinano i backup utilizzando Kopia come data mover, è possibile specificare facoltativamente annotazioni nel CR o tramite la CLI per controllare il comportamento dello storage temporaneo utilizzato da Kopia. Consultare il "Documentazione Kopia" per ulteriori informazioni sulle opzioni che è possibile configurare. Utilizzare il `tridentctl-protect create --help` comando per ulteriori informazioni sulla specifica delle annotazioni con la Trident Protect CLI.

Utilizzare un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-backup-restore-cr.yaml`.
2. Nel file che hai creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono archiviati i contenuti del backup. È possibile utilizzare il seguente comando per trovare questo percorso:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Obbligatorio*) Il nome del AppVault in cui sono archiviati i contenuti del 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**: Il mapping dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituisci `my-source-namespace` e `my-destination-namespace` con le informazioni del tuo 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. (*Facoltativo*) Se è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda le risorse contrassegnate con etichette particolari:

   NOTA Trident Protect seleziona automaticamente alcune risorse in base alla loro relazione con le risorse che selezioni. Ad esempio, se selezioni una risorsa di richiesta di volume persistente e questa ha un pod associato, Trident Protect ripristinerà anche il pod associato.

   - **resourceFilter.resourceSelectionCriteria**: (Obbligatorio per il filtraggio) Utilizzare `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 resourceMatcher oggetti. Se si definiscono più elementi in questo array, la corrispondenza avviene tramite un'operazione OR, e i campi all'interno di ciascun elemento (group, kind, version) corrispondono tramite un'operazione AND.
  - **resourceMatchers[].group**: (*Facoltativo*) Gruppo della risorsa da filtrare.
  - **resourceMatchers[].kind**: (*Facoltativo*) Tipo di risorsa da filtrare.
  - **resourceMatchers[].version**: (*Facoltativo*) Versione della risorsa da filtrare.
  - **resourceMatchers[].names**: (*Facoltativo*) Nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
  - **resourceMatchers[].namespaces**: (*Facoltativo*) Namespace nel campo metadata.name di Kubernetes della risorsa da filtrare.
  - **resourceMatchers[].labelSelectors**: (*Facoltativo*) Stringa del selettore di etichetta nel campo metadata.name dei metadati 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"]
```

4. Dopo aver popolato il `trident-protect-backup-restore-cr.yaml` file con i valori corretti, applica la CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Usa la CLI

Passaggi

1. Ripristina il backup in un namespace diverso, sostituendo i valori tra parentesi con le informazioni del

tuo ambiente. L'argomento `namespace-mapping` utilizza namespace separati da due punti per mappare i namespace di origine ai namespace 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>
```

Ripristina 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 ripristino 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

Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino s3 di lunga durata. Se il token scade durante l'operazione di ripristino, l'operazione può fallire.

- Fare riferimento a "Documentazione API AWS" per ulteriori informazioni sulla verifica della scadenza del token della sessione corrente.
- Fare riferimento a "Documentazione IAM AWS" per ulteriori informazioni sulle credenziali con le risorse AWS.

NOTA

Quando si ripristinano i backup utilizzando Kopia come data mover, è possibile specificare facoltativamente annotazioni nel CR o tramite la CLI per controllare il comportamento dello storage temporaneo utilizzato da Kopia. Consultare il "Documentazione Kopia" per ulteriori informazioni sulle opzioni che è possibile configurare. Utilizzare il `tridentctl-protect create --help` comando per ulteriori informazioni sulla specifica delle annotazioni con la Trident Protect CLI.

Utilizzare un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-backup-ipr-cr.yaml`.
2. Nel file che hai creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono archiviati i contenuti del backup. È possibile utilizzare il seguente comando per trovare questo percorso:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Obbligatorio*) Il nome del AppVault in cui sono archiviati i contenuti del 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. (*Facoltativo*) Se è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda le risorse contrassegnate con etichette particolari:

   NOTA Trident Protect seleziona automaticamente alcune risorse in base alla loro relazione con le risorse che selezioni. Ad esempio, se selezioni una risorsa di richiesta di volume persistente e questa ha un pod associato, Trident Protect ripristinerà anche il pod associato.

   - **resourceFilter.resourceSelectionCriteria**: (Obbligatorio per il filtraggio) Utilizzare `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 resourceMatcher oggetti. Se si definiscono più elementi in questo array, la corrispondenza avviene tramite un'operazione OR, e i campi all'interno di ciascun elemento (group, kind, version) corrispondono tramite un'operazione AND.
       - **resourceMatchers[].group**: (*Facoltativo*) Gruppo della risorsa da filtrare.
       - **resourceMatchers[].kind**: (*Facoltativo*) Tipo di risorsa da filtrare.

- **resourceMatchers[].version**: (*Facoltativo*) Versione della risorsa da filtrare.
- **resourceMatchers[].names**: (*Facoltativo*) Nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].namespaces**: (*Facoltativo*) Namespace nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].labelSelectors**: (*Facoltativo*) Stringa del selettore di etichetta nel campo metadata.name dei metadati 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"]
```

4. Dopo aver popolato il `trident-protect-backup-ipr-cr.yaml` file con i valori corretti, applica la CR:

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

Usa la CLI

Passaggi

1. Ripristina il backup nello spazio dei nomi originale, sostituendo i valori tra parentesi con le informazioni del tuo ambiente. L'argomento `backup` 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>
```

Ripristina da un backup a un cluster diverso

È possibile ripristinare un backup su un cluster diverso se si verifica un problema con il cluster originale.

NOTA

- Quando si ripristinano i backup utilizzando Kopia come data mover, è possibile specificare facoltativamente annotazioni nel CR o tramite la CLI per controllare il comportamento dello storage temporaneo utilizzato da Kopia. Consultare il "Documentazione Kopia" per ulteriori informazioni sulle opzioni che è possibile configurare. Utilizzare il `tridentctl-protect create --help` comando per ulteriori informazioni sulla specifica delle annotazioni con la Trident Protect CLI.
- Quando si utilizza una CR per ripristinare in 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:

- Il cluster di destinazione ha Trident Protect installato.
- Il cluster di destinazione ha accesso al percorso del bucket dello stesso AppVault del cluster di origine, dove è archiviato il backup.
- Assicurati che l'ambiente locale possa connettersi al bucket di storage a oggetti definito nella AppVault CR durante l'esecuzione del comando `tridentctl-protect get appvaultcontent`. Se le restrizioni di rete impediscono l'accesso, esegui la CLI di Trident Protect da un pod sul cluster di destinazione invece.
- Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino di lunga durata. Se il token scade durante l'operazione di ripristino, l'operazione può fallire.
  - Fare riferimento a "Documentazione API AWS" per ulteriori informazioni sulla verifica della scadenza del token della sessione corrente.
  - Fare riferimento a "Documentazione AWS" per ulteriori informazioni sulle credenziali con le risorse AWS.

Passaggi

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 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. Ripristina l’applicazione nel cluster di destinazione utilizzando il nome AppVault e il percorso di archivio:

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. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-backup-restore-cr.yaml`.
2. Nel file che hai creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.appVaultRef**: (*Obbligatorio*) Il nome del AppVault in cui sono archiviati i contenuti del 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**: Il mapping dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituisci `my-source-namespace` e `my-destination-namespace` con le informazioni del tuo 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, applica la CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Usa la CLI

1. Utilizzare il seguente comando per ripristinare l'applicazione, sostituendo i valori tra parentesi con le informazioni del proprio ambiente. L'argomento namespace-mapping utilizza namespace separati da due punti per mappare i namespace di origine ai namespace 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>
```

Ripristina da uno Snapshot a uno spazio dei nomi diverso

È possibile ripristinare i dati da uno snapshot utilizzando un file di risorsa personalizzata (CR) sia in un namespace diverso che nel namespace di origine. Quando si ripristina uno snapshot in un namespace diverso utilizzando una SnapshotRestore CR, Trident Protect ripristina l'applicazione in un nuovo namespace e crea una CR dell'applicazione per l'applicazione ripristinata. Per proteggere l'applicazione ripristinata, crea backup o snapshot on-demand oppure stabilisci una pianificazione di protezione.

NOTA

- SnapshotRestore supporta l' `spec.storageClassMapping` attributo, ma solo quando le classi di archiviazione di origine e destinazione utilizzano lo stesso backend di archiviazione. Se si tenta di eseguire il ripristino su una `StorageClass` che utilizza un backend di archiviazione diverso, l'operazione di ripristino non riuscirà.
- Quando si utilizza una CR per ripristinare in 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 la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino s3 di lunga durata. Se il token scade durante l'operazione di ripristino, l'operazione può fallire.

- Fare riferimento a "Documentazione API AWS" per ulteriori informazioni sulla verifica della scadenza del token della sessione corrente.
- Fare riferimento a "Documentazione IAM AWS" per ulteriori informazioni sulle credenziali con le risorse AWS.

Utilizzare un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-snapshot-restore-cr.yaml`.
2. Nel file che hai creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.appVaultRef**: (*Obbligatorio*) Il nome del AppVault in cui sono archiviati i contenuti dello snapshot.
   - **spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono archiviati i contenuti dello snapshot. È possibile utilizzare il seguente comando per trovare questo percorso:

     ```
     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**: Il mapping dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituisci `my-source-namespace` e `my-destination-namespace` con le informazioni del tuo ambiente.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. (*Facoltativo*) Se è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda le risorse contrassegnate con etichette particolari:

   NOTA Trident Protect seleziona automaticamente alcune risorse in base alla loro relazione con le risorse che selezioni. Ad esempio, se selezioni una risorsa di richiesta di volume persistente e questa ha un pod associato, Trident Protect ripristinerà anche il pod associato.

   - **resourceFilter.resourceSelectionCriteria**: (Obbligatorio per il filtraggio) Utilizzare `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 resourceMatcher oggetti. Se si definiscono più elementi in questo array, la corrispondenza avviene tramite un'operazione OR, e i campi all'interno di ciascun elemento (group, kind, version) corrispondono tramite un'operazione AND.
  - **resourceMatchers[].group**: (*Facoltativo*) Gruppo della risorsa da filtrare.
  - **resourceMatchers[].kind**: (*Facoltativo*) Tipo di risorsa da filtrare.
  - **resourceMatchers[].version**: (*Facoltativo*) Versione della risorsa da filtrare.
  - **resourceMatchers[].names**: (*Facoltativo*) Nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
  - **resourceMatchers[].namespaces**: (*Facoltativo*) Namespace nel campo metadata.name di Kubernetes della risorsa da filtrare.
  - **resourceMatchers[].labelSelectors**: (*Facoltativo*) Stringa del selettore di etichetta nel campo metadata.name dei metadati 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"]
```

4. Dopo aver popolato il `trident-protect-snapshot-restore-cr.yaml` file con i valori corretti, applica la CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Usa la CLI

Passaggi

1. Ripristina l'istantanea in uno spazio dei nomi diverso, sostituendo i valori tra parentesi con le informazioni del tuo ambiente.

- L'`snapshot`argomento utilizza uno spazio dei nomi e un nome di snapshot nel formato `<namespace>/<name>`.
- L'argomento `namespace-mapping` utilizza namespace separati da due punti per mappare i namespace di origine ai namespace 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>
```

Ripristina 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

Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino s3 di lunga durata. Se il token scade durante l'operazione di ripristino, l'operazione può fallire.

- Fare riferimento a "Documentazione API AWS" per ulteriori informazioni sulla verifica della scadenza del token della sessione corrente.
- Fare riferimento a "Documentazione IAM AWS" per ulteriori informazioni sulle credenziali con le risorse AWS.

Utilizzare un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-snapshot-ipr-cr.yaml`.
2. Nel file che hai creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.appVaultRef**: (*Obbligatorio*) Il nome del AppVault in cui sono archiviati i contenuti dello snapshot.
   - **spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono archiviati i contenuti dello snapshot. È possibile utilizzare il seguente comando per trovare questo percorso:

     ```
     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
     ```

3. (*Facoltativo*) Se è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda le risorse contrassegnate con etichette particolari:

   NOTA Trident Protect seleziona automaticamente alcune risorse in base alla loro relazione con le risorse che selezioni. Ad esempio, se selezioni una risorsa di richiesta di volume persistente e questa ha un pod associato, Trident Protect ripristinerà anche il pod associato.

   - **resourceFilter.resourceSelectionCriteria**: (Obbligatorio per il filtraggio) Utilizzare `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 resourceMatcher oggetti. Se si definiscono più elementi in questo array, la corrispondenza avviene tramite un'operazione OR, e i campi all'interno di ciascun elemento (group, kind, version) corrispondono tramite un'operazione AND.
       - **resourceMatchers[].group**: (*Facoltativo*) Gruppo della risorsa da filtrare.
       - **resourceMatchers[].kind**: (*Facoltativo*) Tipo di risorsa da filtrare.
       - **resourceMatchers[].version**: (*Facoltativo*) Versione della risorsa da filtrare.

- **resourceMatchers[].names**: (*Facoltativo*) Nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].namespaces**: (*Facoltativo*) Namespace nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].labelSelectors**: (*Facoltativo*) Stringa del selettore di etichetta nel campo metadata.name dei metadati 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"]
```

4. Dopo aver popolato il `trident-protect-snapshot-ipr-cr.yaml` file con i valori corretti, applica la CR:

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

Usa la CLI

Passaggi

1. Ripristina lo snapshot nello spazio dei nomi originale, sostituendo i valori tra parentesi con le informazioni del tuo ambiente. Ad esempio:

```
tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
-n <application_namespace>
```

Verificare 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.

Passaggi

1. Utilizzare il seguente comando per recuperare lo stato dell'operazione di ripristino, sostituendo i valori tra parentesi con le informazioni dal proprio ambiente:

```
kubectl get backuprestore -n <namespace_name> <my_restore_cr_name> -o
jsonpath='{.status}'
```

Utilizza le impostazioni di ripristino avanzate di Trident Protect

È possibile personalizzare le operazioni di ripristino utilizzando impostazioni avanzate come annotazioni, impostazioni dello spazio dei nomi e opzioni di storage per soddisfare i requisiti specifici.

Annotazioni ed etichette dello spazio dei nomi durante le operazioni di ripristino e failover

Durante le operazioni di ripristino e failover, le etichette e le annotazioni nello spazio dei nomi di destinazione vengono rese corrispondenti alle etichette e alle annotazioni nello spazio dei nomi di origine. Le etichette o le annotazioni dello spazio dei nomi di origine che non esistono nello spazio dei nomi di destinazione vengono aggiunte e tutte le etichette o annotazioni già esistenti vengono sovrascritte per corrispondere al valore dello spazio dei nomi di origine. Le etichette o le annotazioni che esistono 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 e alle configurazioni di sicurezza appropriate definite dai vincoli del contesto di sicurezza (SCC) di OpenShift e possano accedere ai volumi senza problemi di autorizzazione. Per ulteriori informazioni, consulta la "Documentazione sui vincoli del contesto di sicurezza di OpenShift".

È possibile impedire che specifiche annotazioni nello spazio dei nomi di destinazione vengano sovrascritte impostando la variabile di 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 di failover, tutte le annotazioni e le etichette dello spazio dei nomi specificate in `restoreSkipNamespaceAnnotations` e `restoreSkipNamespaceLabels` sono escluse dall'operazione di ripristino o di failover. Assicurarsi che queste impostazioni siano configurate durante l'installazione iniziale di Helm. Per ulteriori informazioni, consultare "Configura impostazioni aggiuntive dell'helm chart Trident Protect".

Se hai installato l'applicazione sorgente utilizzando Helm con il `--create-namespace` flag, viene riservato un trattamento speciale alla chiave dell'etichetta `name`. Durante il processo di ripristino o failover, Trident Protect copia questa etichetta nello spazio dei nomi di destinazione, ma aggiorna il valore a quello dello spazio dei nomi di destinazione se il valore della sorgente 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

Il seguente esempio presenta uno spazio dei nomi di origine e uno di destinazione, ciascuno con annotazioni ed etichette diverse. Puoi vedere lo stato dello spazio dei nomi di destinazione prima e dopo l'operazione e come 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:

Spazio dei nomi	Annotazioni	Etichette
Namespace ns-1 (origine)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• ambiente=produzione • compliance=hipaa • name=ns-1
Namespace ns-2 (destinazione)	• annotation.one/key: "true" • annotazione.tre/chiave: "false"	• role=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, alcune sono state sovrascritte e l' `name`etichetta è stata aggiornata per corrispondere allo spazio dei nomi di destinazione:

Spazio dei nomi	Annotazioni	Etichette
Namespace ns-2 (destinazione)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotazione.tre/chiave: "false"	• name=ns-2 • compliance=hipaa • ambiente=produzione • role=database

Campi supportati

Questa sezione descrive i campi aggiuntivi disponibili per le operazioni di ripristino.

Mappatura delle storage class

L' `spec.storageClassMapping`attributo definisce una mappatura da una classe di storage presente nell'applicazione di origine a una nuova classe di storage nel cluster di destinazione. Puoi utilizzare questa opzione quando migri applicazioni tra cluster con classi di storage diverse o quando cambi il backend di storage per le operazioni di 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
protect.trident.netapp.io/data-mover-timeout-sec	stringa	Il tempo massimo (in secondi) consentito per l'operazione di spostamento dei dati che può essere bloccata.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	stringa	Limite massimo di dimensione (in megabyte) per la cache dei contenuti Kopia.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	stringa	Tempo massimo (in secondi) di attesa affinché eventuali nuove PersistentVolumeClaims (PVC) raggiungano la fase `Bound` prima che l'operazione fallisca. Si applica a tutti i tipi di restore CR (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Utilizzare un valore più alto se il backend di storage o il cluster richiede spesso più tempo.	"1200" (20 minuti)

Replicare le applicazioni utilizzando NetApp SnapMirror e Trident Protect

Utilizzando Trident Protect, puoi utilizzare le capacità di replica asincrona della tecnologia NetApp SnapMirror per replicare dati e modifiche delle applicazioni da un backend di storage a un altro, sullo stesso cluster o tra cluster diversi.

Annotazioni ed etichette dello spazio dei nomi durante le operazioni di ripristino e failover

Durante le operazioni di ripristino e failover, le etichette e le annotazioni nello spazio dei nomi di destinazione vengono rese corrispondenti alle etichette e alle annotazioni nello spazio dei nomi di origine. Le etichette o le annotazioni dello spazio dei nomi di origine che non esistono nello spazio dei nomi di destinazione vengono aggiunte e tutte le etichette o annotazioni già esistenti vengono sovrascritte per corrispondere al valore dello spazio dei nomi di origine. Le etichette o le annotazioni che esistono 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 e alle configurazioni di sicurezza appropriate definite dai vincoli del contesto di sicurezza (SCC) di OpenShift e possano accedere ai volumi senza problemi di autorizzazione. Per ulteriori informazioni, consulta la "Documentazione sui vincoli del contesto di sicurezza di OpenShift".

È possibile impedire che specifiche annotazioni nello spazio dei nomi di destinazione vengano sovrascritte impostando la variabile di 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_key_to_skip_2>}" \
  --reuse-values
```

NOTA Quando si esegue un'operazione di ripristino o di failover, tutte le annotazioni e le etichette dello spazio dei nomi specificate in `restoreSkipNamespaceAnnotations` e `restoreSkipNamespaceLabels` sono escluse dall'operazione di ripristino o di failover. Assicurarsi che queste impostazioni siano configurate durante l'installazione iniziale di Helm. Per ulteriori informazioni, consultare "Configura impostazioni aggiuntive dell'helm chart Trident Protect".

Se hai installato l'applicazione sorgente utilizzando Helm con il `--create-namespace` flag, viene riservato un trattamento speciale alla chiave dell'etichetta `name`. Durante il processo di ripristino o failover, Trident Protect copia questa etichetta nello spazio dei nomi di destinazione, ma aggiorna il valore a quello dello spazio dei nomi di destinazione se il valore della sorgente 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

Il seguente esempio presenta uno spazio dei nomi di origine e uno di destinazione, ciascuno con annotazioni ed etichette diverse. Puoi vedere lo stato dello spazio dei nomi di destinazione prima e dopo l'operazione e come 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:

Spazio dei nomi	Annotazioni	Etichette
Namespace ns-1 (origine)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• ambiente=produzione • compliance=hipaa • name=ns-1
Namespace ns-2 (destinazione)	• annotation.one/key: "true" • annotazione.tre/chiave: "false"	• role=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, alcune sono state sovrascritte e l' `name`etichetta è stata aggiornata per corrispondere allo spazio dei nomi di destinazione:

Spazio dei nomi	Annotazioni	Etichette
Namespace ns-2 (destinazione)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotazione.tre/chiave: "false"	• name=ns-2 • compliance=hipaa • ambiente=produzione • role=database

NOTA È possibile configurare Trident Protect per congelare e scongelare i filesystem durante le operazioni di protezione dei dati. "Scopri di più sulla configurazione del congelamento del filesystem con Trident Protect".

Hook di esecuzione durante le operazioni di failover e reverse

Quando si utilizza una relazione AppMirror per proteggere l'applicazione, è necessario essere a conoscenza di comportamenti specifici relativi agli hook di esecuzione durante le operazioni di failover e reverse.

- Durante il failover, gli hook di esecuzione vengono copiati automaticamente dal cluster di origine al cluster di destinazione. Non è necessario ricrearli manualmente. Dopo il failover, gli hook di esecuzione sono presenti sull'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 execution hook, fare riferimento a "Gestire gli hook di esecuzione di Trident Protect".

Imposta una relazione di replica

L'impostazione di una relazione di replicazione 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 così come gli snapshot del volume per ciascuno dei volumi dell'app)
- Scelta della pianificazione della replica (include risorse Kubernetes e dati di volume persistente)

- Impostazione dell’ora in cui verrà scattata l’istantanea

Passaggi

1. Nel cluster di origine, crea un AppVault per l’applicazione di origine. A seconda del provider di storage, modificare un esempio in "Risorse personalizzate AppVault" per adattarlo al proprio ambiente:

Crea un AppVault utilizzando un CR

a. Creare il file custom resource (CR) e assegnargli un nome (ad esempio `trident-protect-appvault-primary-source.yaml`).

b. Configura i seguenti attributi:

- **metadata.name**: (*Obbligatorio*) Il nome della risorsa personalizzata AppVault. Prendi nota del nome che scegli, perché altri file CR necessari per una relazione di replica fanno riferimento a questo valore.
- **spec.providerConfig**: (*Obbligatorio*) Memorizza la configurazione necessaria per accedere a AppVault utilizzando il provider specificato. Scegli un bucketName e qualsiasi altro dettaglio necessario per il tuo provider. Prendi nota dei valori che scegli, perché altri file CR necessari per una relazione di replica fanno riferimento a questi valori. Consulta "Risorse personalizzate AppVault" per esempi di CR AppVault con altri provider.
- **spec.providerCredentials**: (*Obbligatorio*) Memorizza i riferimenti a qualsiasi credenziale richiesta per accedere al AppVault utilizzando il provider specificato.
  - **spec.providerCredentials.valueFromSecret**: (*Obbligatorio*) Indica che il valore delle credenziali deve provenire da un segreto.
    - **key**: (*Obbligatorio*) La chiave valida del segreto da selezionare.
    - **name**: (*Obbligatorio*) Nome del secret contenente il valore per questo campo. Deve essere nello stesso namespace.
  - **spec.providerCredentials.secretAccessKey**: (*Obbligatorio*) La chiave di accesso utilizzata per accedere al provider. Il **nome** deve corrispondere a **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*Obbligatorio*) Determina cosa fornisce il backup; ad esempio, NetApp ONTAP S3, S3 generico, Google Cloud o Microsoft Azure. Valori possibili:
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. Dopo aver popolato il `trident-protect-appvault-primary-source.yaml` file con i valori corretti, applica la CR:

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n
trident-protect
```

Creare un AppVault utilizzando la CLI

a. Crea il AppVault, sostituendo i valori tra parentesi con le informazioni del tuo 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 la CR dell'applicazione di origine:

Crea l'applicazione di origine utilizzando un CR

a. Creare il file custom resource (CR) e assegnargli un nome (ad esempio `trident-protect-app-source.yaml`).

b. Configura i seguenti attributi:

- **metadata.name**: (*Required*) Il nome della risorsa personalizzata dell'applicazione. Prendi nota del nome che scegli, perché 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. Usare i nomi degli spazi dei nomi e, facoltativamente, restringere l'ambito degli spazi dei nomi con le etichette per specificare le risorse che esistono negli spazi dei nomi qui elencati. 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, applica la CR:

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-namespace
```

Crea l'applicazione sorgente utilizzando la CLI

a. Crea 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, acquisire un'istantanea dell'applicazione di origine. Questa istantanea viene utilizzata come base per l'applicazione sul cluster di destinazione. Se si salta questo passaggio, sarà necessario attendere l'esecuzione della prossima istantanea programmata per avere un'istantanea recente. Per creare un'istantanea su richiesta, fare riferimento a "Crea un'istantanea su richiesta".

4. Nel cluster di origine, creare la pianificazione di replica CR:

Oltre alla pianificazione fornita di seguito, si consiglia di creare una pianificazione separata delle snapshot giornaliere con un periodo di conservazione di 7 giorni per mantenere una snapshot comune tra i cluster ONTAP peered. Questo garantisce che le snapshot siano disponibili per un massimo di 7 giorni, ma il periodo di conservazione può essere personalizzato in base alle esigenze degli utenti.

NOTA Se si verifica un failover, il sistema può utilizzare queste snapshot per un massimo di 7 giorni per le operazioni di inversione. Questo approccio rende il processo di inversione più rapido ed efficiente perché vengono trasferite solo le modifiche apportate dall'ultima snapshot, non tutti i dati.

Se una pianificazione esistente per l'applicazione soddisfa già i requisiti di conservazione desiderati, non sono necessarie ulteriori pianificazioni.

Crea il programma di replica utilizzando un CR

a. Crea una pianificazione di replica per l'applicazione di origine:

   i. Creare il file custom resource (CR) e assegnargli un nome (ad esempio `trident-protect-schedule.yaml`).

   ii. Configura i seguenti attributi:

   - **metadata.name**: (*Richiesta*) Il nome della risorsa personalizzata della pianificazione.
   - **spec.appVaultRef**: (*Required*) Questo valore deve corrispondere al campo metadata.name di AppVault per l'applicazione di origine.
   - **spec.applicationRef**: (*Required*) Questo valore deve corrispondere al campo metadata.name dell'applicazione CR 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**: Definire una data di inizio in 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, applica la CR:

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-namespace
```

Crea il programma di replica utilizzando la CLI

a. Crea il programma di replica, sostituendo i valori tra parentesi con le informazioni del 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. Sul cluster di destinazione, crea una source application AppVault CR identica alla AppVault CR che hai applicato sul cluster di origine e assegnale un nome (ad esempio, `trident-protect-appvault-primary-destination.yaml`).
6. Applica il CR:

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n
trident-protect
```

7. Creare un AppVault CR di destinazione per l'applicazione di destinazione sul cluster di destinazione. A seconda del provider di storage, modificare un esempio in "Risorse personalizzate AppVault" per adattarlo al proprio ambiente:
   a. Creare il file custom resource (CR) e assegnargli un nome (ad esempio `trident-protect-appvault-secondary-destination.yaml`).
   b. Configura i seguenti attributi:
      - **metadata.name**: (*Obbligatorio*) Il nome della risorsa personalizzata AppVault. Prendi nota del nome che scegli, perché altri file CR necessari per una relazione di replica fanno riferimento a questo valore.
      - **spec.providerConfig**: (*Obbligatorio*) Memorizza la configurazione necessaria per accedere a AppVault utilizzando il provider specificato. Scegli un `bucketName` e qualsiasi altro dettaglio necessario per il tuo provider. Prendi nota dei valori che scegli, perché altri file CR necessari per una relazione di replica fanno riferimento a questi valori. Consulta "Risorse personalizzate AppVault" per esempi di CR AppVault con altri provider.
      - **spec.providerCredentials**: (*Obbligatorio*) Memorizza i riferimenti a qualsiasi credenziale richiesta per accedere al AppVault utilizzando il provider specificato.
        - **spec.providerCredentials.valueFromSecret**: (*Obbligatorio*) Indica che il valore delle

credenziali deve provenire da un segreto.

- **key**: (*Obbligatorio*) La chiave valida del segreto da selezionare.
- **name**: (*Obbligatorio*) Nome del secret contenente il valore per questo campo. Deve essere nello stesso namespace.

- **spec.providerCredentials.secretAccessKey**: (*Obbligatorio*) La chiave di accesso utilizzata per accedere al provider. Il **nome** deve corrispondere a **spec.providerCredentials.valueFromSecret.name**.

- **spec.providerType**: (*Obbligatorio*) Determina cosa fornisce il backup; ad esempio, NetApp ONTAP S3, S3 generico, Google Cloud o Microsoft Azure. Valori possibili:
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. Dopo aver popolato il `trident-protect-appvault-secondary-destination.yaml` file con i valori corretti, applica la CR:

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml
-n trident-protect
```

8. Sul cluster di destinazione, creare un file CR AppMirrorRelationship.

NOTA Quando si utilizza un CR, creare manualmente lo spazio dei nomi della destinazione prima di applicare il CR. Trident Protect crea automaticamente gli spazi dei nomi solo quando si utilizza la CLI.

Crea un AppMirrorRelationship utilizzando un CR

a. Creare il file custom resource (CR) e assegnargli un nome (ad esempio `trident-protect-relationship.yaml`).

b. Configura i seguenti attributi:

- **metadata.name:** (Obbligatorio) Il nome della risorsa personalizzata AppMirrorRelationship.
- **spec.destinationAppVaultRef**: (*Required*) Questo valore deve corrispondere al nome di AppVault per l'applicazione di destinazione sul cluster di destinazione.
- **spec.namespaceMapping**: (*Obbligatorio*) Gli spazi dei nomi di destinazione e di origine devono corrispondere allo spazio dei nomi dell'applicazione definito nel rispettivo CR dell'applicazione.
- **spec.sourceAppVaultRef**: (*Required*) Questo valore deve corrispondere al nome del AppVault per l'applicazione di origine.
- **spec.sourceApplicationName**: (*Obbligatorio*) Questo valore deve corrispondere al nome dell'applicazione di origine che hai definito nel CR dell'applicazione di origine.
- **spec.sourceApplicationUID**: (Obbligatorio) Questo valore deve corrispondere all'UID dell'applicazione di origine che hai definito nel CR dell'applicazione di origine.
- **storageClassName**: (*Opzionale*) Scegli il nome di una classe di archiviazione valida sul cluster. La classe di archiviazione deve essere collegata a una VM di archiviazione ONTAP che è peered con l'ambiente di origine. Se la classe di archiviazione non viene fornita, la classe di archiviazione predefinita sul cluster verrà utilizzata per impostazione predefinita.
- **spec.recurrenceRule**: Definire una data di inizio in 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, applica la CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-
namespace
```

Creare un AppMirrorRelationship utilizzando la CLI

a. Crea e applica l'oggetto AppMirrorRelationship, sostituendo i valori tra parentesi con le informazioni del 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> -n <application_namespace>
```

Esempio:

```
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. (*Opzionale*) Sul cluster di destinazione, verificare lo stato e la situazione della relazione di replica:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Fail over verso il cluster di destinazione

Utilizzando Trident Protect, è possibile effettuare 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 interrompe l'app sul cluster di origine se era operativa.

Passaggi

1. Sul cluster di destinazione, modificare il file CR AppMirrorRelationship (ad esempio, `trident-protect-relationship.yaml`) e cambiare il valore di **spec.desiredState** in `Promoted`.
2. Salva il file CR.
3. Applica il CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. (*Opzionale*) Crea tutte le policy di protezione di cui hai bisogno sull'applicazione su cui è stato effettuato il failover.
5. (*Opzionale*) Controlla lo stato e lo stato della relazione di replica:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Risincronizza una relazione di replica sottoposta a failover

L'operazione di risincronizzazione ristabilisce la relazione di replica. Dopo aver eseguito un'operazione di risincronizzazione, l'applicazione di origine originale diventa l'applicazione in esecuzione e tutte le modifiche apportate all'applicazione in esecuzione sul cluster di destinazione sono 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.

Passaggi

1. Opzionale: Sul cluster di origine, crea uno snapshot dell'applicazione di origine. Questo assicura che le ultime modifiche dal cluster di origine vengano acquisite.
2. Sul cluster di destinazione, modificare il file CR AppMirrorRelationship (ad esempio, `trident-protect-relationship.yaml`) e cambiare il valore di spec.desiredState in `Established`.
3. Salva il file CR.
4. Applica il CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. Se sono state create pianificazioni di protezione sul cluster di destinazione per proteggere l'applicazione sottoposta a failover, rimuoverle. Eventuali pianificazioni rimaste causano errori di snapshot del volume.

Risincronizza inversamente una relazione di replica sottoposta a failover

Quando si esegue la risincronizzazione inversa di una relazione di replica fallita, 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.

Passaggi

1. Sul cluster di destinazione originale, eliminare il CR AppMirrorRelationship. Questo fa sì che la destinazione diventi la sorgente. Se ci sono policy di protezione rimanenti sul nuovo cluster di destinazione, rimuoverle.
2. Imposta una relazione di replica applicando i file CR che hai utilizzato originariamente per impostare la relazione ai cluster opposti.
3. Assicurarsi che la nuova destinazione (cluster di origine originale) sia configurata con entrambi i CR AppVault.
4. Imposta 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 di replica, Trident Protect sposta l'applicazione sul backend di storage di destinazione continuando a replicare verso il backend di storage di origine. Trident Protect arresta l'applicazione di origine e replica i dati sulla destinazione prima di passare all'applicazione di destinazione.

In questa situazione, si scambiano il cluster di origine e il cluster di destinazione.

Passaggi

1. Sul cluster di origine, creare una snapshot di spegnimento:

Crea un'istantanea di spegnimento utilizzando un CR

a. Disattivare le pianificazioni della policy di protezione per l'applicazione di origine.

b. Crea un file ShutdownSnapshot CR:

i. Creare il file custom resource (CR) e assegnargli un nome (ad esempio `trident-protect-shutdownsnapshot.yaml`).

ii. Configura i seguenti attributi:

- **metadata.name**: (*Richiesto*) Il nome della risorsa personalizzata.
- **spec.AppVaultRef**: (*Obbligatorio*) Questo valore deve corrispondere al campo metadata.name di 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, applica la CR:

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-
namespace
```

Crea una Snapshot di spegnimento utilizzando la CLI

a. Crea la Snapshot di spegnimento, sostituendo i valori tra parentesi con le informazioni del tuo ambiente. Ad esempio:

```
tridentctl-protect create shutdownsnapshot <my_shutdown_snapshot>
--appvault <my_vault> --app <app_to_snapshot> -n
<application_namespace>
```

2. Nel cluster di origine, dopo il completamento della Snapshot di spegnimento, ottenere lo stato della Snapshot di spegnimento:

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. Sul cluster di origine, trovare il valore di **shutdownsnapshot.status.appArchivePath** usando il seguente comando e registrare l'ultima parte del percorso del file (chiamata anche basename; sarà tutto ciò che si trova dopo l'ultima barra):

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o
jsonpath='{.status.appArchivePath}'
```

4. Eseguire un fail over dal nuovo cluster di destinazione al nuovo cluster di origine, con la seguente modifica:

NOTA	Nel passo 2 della procedura di fail over, includere il `spec.promotedSnapshot` campo nel file AppMirrorRelationship CR e impostare il suo valore sul basename registrato nel passo 3 sopra.

5. Eseguire i passaggi di risincronizzazione inversa in Risincronizza inversamente una relazione di replica sottoposta a failover.
6. Abilitare le pianificazioni di protezione sul nuovo cluster di origine.

Risultato

Le seguenti azioni si verificano a causa della replica inversa:

- Viene scattata una Snapshot delle risorse Kubernetes dell'app di origine.
- I pod dell'app originale vengono interrotti in modo graduale eliminando le risorse Kubernetes dell'app (lasciando PVC e PV al loro posto).
- Dopo lo spegnimento dei pod, vengono acquisite e replicate le Snapshot dei volumi dell'applicazione.
- Le relazioni SnapMirror vengono interrotte, rendendo i volumi di destinazione pronti per la lettura/scrittura.
- Le risorse Kubernetes dell'app vengono ripristinate dalla snapshot pre-arresto, utilizzando i dati del volume replicati dopo che l'app di origine originale è stata arrestata.
- La replicazione viene ristabilita in senso inverso.

Ripristina le applicazioni nel cluster di origine

Utilizzando Trident Protect, puoi 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 replica originale, Trident Protect replica (risincronizza) eventuali modifiche dell'applicazione all'applicazione sorgente originale prima di invertire la direzione di replica.

Questo processo inizia da una relazione che ha completato un failover verso una destinazione e comporta i seguenti passaggi:

- Inizia con uno stato di failover.
- Invertire la risincronizzazione della relazione di replica.

AVVERTENZA	Non eseguire un'operazione di risincronizzazione normale, perché in questo modo si scartano i dati scritti sul cluster di destinazione durante la procedura di fail over.

- Invertire la direzione di replica.

Passaggi

1. Eseguire i Risincronizza inversamente una relazione di replica sottoposta a failover passaggi.
2. Eseguire i Invertire la direzione di replica dell'applicazione passaggi.

Eliminare una relazione di replica

È possibile eliminare una relazione di replica in qualsiasi momento. Quando si elimina la relazione di replica dell'applicazione, si ottengono due applicazioni separate senza alcuna relazione tra loro.

Passaggi

1. Sul cluster di destinazione corrente, eliminare il CR AppMirrorRelationship:

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

Migra le applicazioni utilizzando Trident Protect

È possibile migrare le applicazioni tra cluster o in classi di storage diverse ripristinando i dati di backup.

NOTA	Quando si esegue la migrazione di un'applicazione, tutti gli execution hook configurati per l'applicazione vengono migrati insieme all'app. Se è presente un execution hook 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 specifiche attività di backup e ripristino.

Clona nello stesso cluster

Per clonare un'applicazione nello stesso cluster, crea uno snapshot o esegui il backup e ripristina i dati nello stesso cluster.

Passaggi

1. Esegui una delle seguenti operazioni:
   a. "Crea uno snapshot".
   b. "Crea un backup".
2. Sullo stesso cluster, esegui una delle seguenti operazioni, a seconda che tu abbia creato uno snapshot o un backup:

a. "Ripristina i tuoi dati dallo snapshot".

b. "Ripristina i tuoi dati dal backup".

Clona in un cluster diverso

Per clonare un'applicazione su un cluster diverso (eseguire un clone tra cluster), crea un backup sul cluster di origine e poi ripristina il backup su un cluster di destinazione. Assicurarsi che Trident Protect sia installato sul cluster di destinazione.

NOTA	È possibile replicare un'applicazione tra cluster diversi utilizzando "Replica SnapMirror".

Passaggi

1. "Crea un backup".
2. Assicurarsi che il CR AppVault per il bucket di storage a oggetti che contiene il backup sia stato configurato sul cluster di destinazione.
3. Sul cluster di destinazione, "ripristina i dati dal backup".

Migrare le applicazioni da una storage class a un'altra storage class

È possibile migrare le applicazioni da una classe di storage a un'altra classe di storage ripristinando un backup nella classe di storage di destinazione.

Ad esempio (escludendo i segreti dal restore CR):

```
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
```

Ripristina l'istantanea utilizzando un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-snapshot-restore-cr.yaml`.
2. Nel file che hai creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono archiviati i contenuti dello snapshot. È possibile utilizzare il seguente comando per trovare questo percorso:

     ```
     kubectl get snapshots <my-snapshot-name> -n trident-protect -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Obbligatorio*) Il nome del AppVault in cui sono archiviati i contenuti dello snapshot.
   - **spec.namespaceMapping**: Il mapping dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituisci `my-source-namespace` e `my-destination-namespace` con le informazioni del tuo 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. Facoltativamente, se è necessario selezionare solo alcune risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda le risorse contrassegnate da particolari etichette:
   - **resourceFilter.resourceSelectionCriteria**: (Richiesto per il filtraggio) Usa `include or 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 resourceMatcher oggetti. Se si definiscono più elementi in questo array, la corrispondenza avviene tramite un'operazione OR, e i campi all'interno di ciascun elemento (group, kind, version) corrispondono tramite un'operazione AND.
       - **resourceMatchers[].group**: (*Facoltativo*) Gruppo della risorsa da filtrare.
       - **resourceMatchers[].kind**: (*Facoltativo*) Tipo di risorsa da filtrare.

- **resourceMatchers[].version**: (*Facoltativo*) Versione della risorsa da filtrare.
- **resourceMatchers[].names**: (*Facoltativo*) Nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].namespaces**: (*Facoltativo*) Namespace nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].labelSelectors**: (*Facoltativo*) Stringa del selettore di etichetta nel campo metadata.name dei metadati 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"]
```

4. Dopo aver popolato il `trident-protect-snapshot-restore-cr.yaml` file con i valori corretti, applica la CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Ripristina l'istantanea utilizzando la CLI

Passaggi

1. Ripristina l'istantanea in uno spazio dei nomi diverso, sostituendo i valori tra parentesi con le informazioni del tuo ambiente.

   - L' `snapshot`argomento utilizza uno spazio dei nomi e un nome di snapshot nel formato `<namespace>/<name>`.
   - L'argomento `namespace-mapping` utilizza namespace separati da due punti per mappare i namespace di origine ai namespace 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>
```

Gestire gli hook di esecuzione di Trident Protect

Un hook di esecuzione è un'azione personalizzata che puoi configurare per essere eseguita insieme a un'operazione di protezione dei dati di un'app gestita. Ad esempio, se hai un'app di database, puoi utilizzare un hook di esecuzione per mettere in pausa tutte le transazioni del database prima di uno Snapshot e riprenderle dopo che lo Snapshot è stato completato. Questo 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 di hook di esecuzione si verificano nel seguente ordine:

1. Tutti gli hook di esecuzione pre-operazione personalizzati applicabili vengono eseguiti sui container appropriati. È possibile creare ed eseguire tutti gli hook di esecuzione pre-operazione personalizzati necessari, ma l'ordine di esecuzione di questi hook prima dell'operazione non è né garantito né configurabile.
2. Si verificano blocchi del file system, se applicabile. "Scopri di più sulla configurazione del congelamento del filesystem con Trident Protect".
3. L'operazione di protezione dei dati viene eseguita.
4. I file system congelati vengono scongelati, se applicabile.
5. Tutti gli hook di esecuzione post-operazione personalizzati applicabili vengono eseguiti sui container appropriati. È possibile creare ed eseguire tutti gli hook di esecuzione post-operazione personalizzati necessari, ma l'ordine di esecuzione di questi hook dopo l'operazione non è né 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, l'ordine di esecuzione di hook di tipi diversi è garantito. Ad esempio, il seguente è l'ordine di esecuzione di una configurazione che ha tutti i diversi tipi di hook:

1. Hook pre-snapshot eseguiti
2. Hook post-snapshot eseguiti
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	Dovresti sempre testare i tuoi script di hook di esecuzione prima di abilitarli in un ambiente di produzione. Puoi utilizzare il comando 'kubectl exec' per testare comodamente gli script. Dopo aver abilitato gli hook di esecuzione in un ambiente di produzione, testa gli Snapshot e i backup risultanti per assicurarti che siano coerenti. Puoi farlo clonando l'app in uno spazio dei nomi temporaneo, ripristinando lo Snapshot o il backup e quindi testando l'app.

NOTA	Se un hook di esecuzione pre-snapshot aggiunge, modifica o rimuove risorse Kubernetes, tali modifiche vengono incluse nello Snapshot o nel backup e in qualsiasi successiva operazione di ripristino.

Note importanti sui ganci di esecuzione personalizzati

Considera quanto segue quando pianifichi gli hook di esecuzione per le tue app.

- Un hook di esecuzione deve utilizzare uno script per eseguire 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 di script shell eseguibili.
- La dimensione dello script è limitata a 96KB.
- Trident Protect utilizza le impostazioni dell'execution hook e qualsiasi criterio corrispondente per determinare quali hook sono applicabili a un'operazione di Snapshot, backup o ripristino.

NOTA	Poiché gli hook di esecuzione spesso riducono o disabilitano completamente la funzionalità dell'applicazione su cui vengono eseguiti, dovresti sempre cercare di ridurre al minimo il tempo necessario per l'esecuzione degli hook di esecuzione personalizzati. Se avvii un'operazione di backup o Snapshot con gli hook di esecuzione associati ma poi la annulli, gli hook sono comunque autorizzati a essere eseguiti se l'operazione di backup o Snapshot è già iniziata. Ciò significa che la logica utilizzata in un hook di esecuzione post-backup non può presumere che il backup sia stato completato.

Filtri hook di esecuzione

Quando si aggiunge o si modifica un hook di esecuzione per un'applicazione, è possibile aggiungere filtri all'hook di esecuzione per gestire quali container corrisponderanno all'hook. I filtri sono utili per le applicazioni che utilizzano la stessa immagine container su tutti i container, ma potrebbero utilizzare ciascuna immagine per uno scopo diverso (ad esempio Elasticsearch). I filtri consentono di creare scenari in cui gli hook di esecuzione vengono eseguiti su alcuni, ma non necessariamente su tutti i container identici. Se si creano più filtri per un singolo hook di esecuzione, questi vengono combinati con un operatore logico AND. È possibile avere fino a 10 filtri attivi per ogni hook di esecuzione.

Ogni filtro che aggiungi a un hook di esecuzione utilizza un'espressione regolare per trovare corrispondenze

con i container nel tuo cluster. Quando un hook trova una corrispondenza con un container, eseguirà lo script associato su quel container. Le espressioni regolari per i filtri utilizzano la sintassi Regular Expression 2 (RE2), che non supporta la creazione di un filtro che escluda i container dall'elenco delle corrispondenze. Per informazioni sulla sintassi che Trident Protect supporta per le espressioni regolari nei filtri degli hook di esecuzione, vedere "Supporto della sintassi Regular Expression 2 (RE2)".

NOTA	Se si aggiunge un filtro namespace a un hook di esecuzione eseguito dopo un'operazione di ripristino o clonazione e l'origine e la destinazione si trovano in namespace diversi, il filtro namespace viene applicato solo al namespace di destinazione.

Esempi di execution hook

Visita il "Progetto NetApp Verda GitHub" per scaricare hook di esecuzione reali per app popolari come Apache Cassandra e Elasticsearch. Puoi anche vedere esempi e ottenere idee per strutturare i tuoi hook di esecuzione personalizzati.

Crea un hook 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 Owner, Admin o Member.

Utilizzare un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-hook.yaml`.
2. Configura i seguenti attributi in modo che corrispondano al tuo ambiente Trident Protect e alla configurazione del cluster:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.applicationRef**: (*Obbligatorio*) Il nome Kubernetes dell'applicazione per cui eseguire l'hook di esecuzione.
   - **spec.stage**: (*Obbligatorio*) Una stringa che indica in quale fase dell'azione deve essere eseguito l'hook di esecuzione. Valori possibili:
     - Pre
     - Post
   - **spec.action**: (*Obbligatorio*) Una stringa che indica quale azione verrà intrapresa dall'hook di esecuzione, supponendo che eventuali filtri dell'hook di esecuzione specificati corrispondano. Valori possibili:
     - Snapshot
     - Backup
     - Ripristina
     - Failover
   - **spec.enabled**: (*Facoltativo*) Indica se questo hook di esecuzione è abilitato o disabilitato. Se non specificato, il valore predefinito è true.
   - **spec.hookSource**: (*Obbligatorio*) Una stringa contenente lo script hook codificato in base64.
   - **spec.timeout**: (*Facoltativo*) Un numero che definisce per quanti minuti è consentita l'esecuzione dell'hook di esecuzione. Il valore minimo è 1 minuto e il valore predefinito è 25 minuti se non specificato.
   - **spec.arguments**: (*Facoltativo*) Un elenco YAML di argomenti che puoi specificare per l'execution hook.
   - **spec.matchingCriteria**: (*Facoltativo*) Un elenco facoltativo di coppie chiave-valore di criteri, ciascuna delle quali costituisce un filtro di hook di esecuzione. È possibile aggiungere fino a 10 filtri per hook di esecuzione.
   - **spec.matchingCriteria.type**: (*Facoltativo*) Una stringa che identifica il tipo di filtro dell'hook di esecuzione. Valori possibili:
     - ContainerImage
     - ContainerName
     - PodName
     - PodLabel
     - NamespaceName
   - **spec.matchingCriteria.value**: (*Facoltativo*) Una stringa o 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, applica il CR:

```
kubectl apply -f trident-protect-hook.yaml
```

Usa la CLI

Passaggi

1. Crea l'hook di esecuzione, sostituendo i valori tra parentesi con le informazioni dal tuo 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>
```

Esegui manualmente un hook di esecuzione

È possibile eseguire manualmente un hook di esecuzione a scopo di test o se è necessario rieseguire manualmente l'hook dopo un errore. È necessario disporre delle autorizzazioni di Owner, Admin o Member per eseguire manualmente gli hook di esecuzione.

L'esecuzione manuale di un hook di esecuzione consiste in due passaggi fondamentali:

1. Crea un backup delle risorse, che raccoglie le risorse e crea un backup di esse, determinando dove verrà eseguito l'hook
2. Esegui l'hook di esecuzione sul backup

Passaggio 1: crea un backup delle risorse

Utilizzare un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-resource-backup.yaml`.
2. Configura i seguenti attributi in modo che corrispondano al tuo ambiente Trident Protect e alla configurazione del cluster:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.applicationRef**: (*Obbligatorio*) Il nome Kubernetes dell'applicazione per cui creare il backup delle risorse.
   - **spec.appVaultRef**: (*Obbligatorio*) Il nome del AppVault in cui sono archiviati i contenuti del backup.
   - **spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono archiviati i contenuti del backup. È possibile utilizzare il seguente comando per trovare questo percorso:

     ```
     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, applica il CR:

```
kubectl apply -f trident-protect-resource-backup.yaml
```

Usa la CLI

Passaggi

1. Crea il backup, sostituendo i valori tra parentesi con le informazioni del tuo 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. Visualizza lo stato del backup. Puoi usare 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 andato a buon fine:

```
kubectl describe resourcebackup <my_backup_name>
```

Passaggio 2: eseguire l'hook di esecuzione

Utilizzare un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-hook-run.yaml`.
2. Configura i seguenti attributi in modo che corrispondano al tuo ambiente Trident Protect e alla configurazione del cluster:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.applicationRef**: (*Obbligatorio*) Assicurati che questo valore corrisponda al nome dell'applicazione dalla ResourceBackup CR che hai creato nel passaggio 1.
   - **spec.appVaultRef**: (*Obbligatorio*) Assicurati che questo valore corrisponda a appVaultRef dal ResourceBackup CR che hai creato nel passaggio 1.
   - **spec.appArchivePath**: assicurati che questo valore corrisponda a appArchivePath dal ResourceBackup CR che hai creato nel passaggio 1.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.action**: (*Obbligatorio*) Una stringa che indica quale azione verrà intrapresa dall'hook di esecuzione, supponendo che eventuali filtri dell'hook di esecuzione specificati corrispondano. Valori possibili:
     - Snapshot
     - Backup
     - Ripristina
     - Failover
   - **spec.stage**: (*Obbligatorio*) Una stringa che indica in quale fase dell'azione deve essere eseguito l'hook di esecuzione. Questa esecuzione dell'hook non eseguirà hook in nessun'altra fase. 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, applica il CR:

```
kubectl apply -f trident-protect-hook-run.yaml
```

Usa la CLI

Passaggi

1. Crea la richiesta di esecuzione manuale dell'hook:

```
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. Controlla lo stato dell'esecuzione del hook. Puoi eseguire questo comando ripetutamente fino al completamento dell'operazione:

```
tridentctl protect get exechooksrun -n <my_app_namespace>
<my_exec_hook_run_name>
```

3. Descrivi l'oggetto exechooksrun per vedere i dettagli finali e lo stato:

```
kubectl -n <my_app_namespace> describe exechooksrun
<my_exec_hook_run_name>
```

Disinstallare Trident Protect

Potrebbe essere necessario rimuovere i componenti di Trident Protect se si esegue l'aggiornamento da una versione di prova a una versione completa del prodotto.

Per rimuovere Trident Protect, eseguire i seguenti passaggi.

Passaggi

1. Rimuovere i file CR di Trident Protect:

   NOTA Questo passaggio non è necessario per la versione 25.06 e successive.

   ```
   helm uninstall -n trident-protect trident-protect-crds
   ```

2. Rimuovi Trident Protect:

   ```
   helm uninstall -n trident-protect trident-protect
   ```

3. Rimuovere lo spazio dei nomi Trident Protect:

   ```
   kubectl delete ns trident-protect
   ```

Blog di Trident e Trident Protect

Qui potete trovare alcuni ottimi blog NetApp Trident e Trident Protect:

Blog di Trident

- 16 ottobre 2025: "Soluzioni di storage avanzate per Kubernetes"
- 19 agosto 2025: "Migliorare la consistenza dei dati: istantanee di gruppi di volumi nella virtualizzazione OpenShift con Trident"
- 09 maggio 2025: "Configurazione automatica del backend Trident per FSx for ONTAP con il componente aggiuntivo Amazon EKS"
- 15 aprile 2025: "NetApp Trident con Google Cloud NetApp Volumes per il protocollo SMB"
- 14 aprile 2025: "Sfruttare il protocollo Fiber Channel con Trident 25.02 per lo storage persistente su Kubernetes"
- 14 aprile 2025: "Liberare la potenza dei sistemi NetApp ASA r2 per lo storage a blocchi Kubernetes"
- 31 marzo 2025: "Semplificazione dell'installazione di Trident su Red Hat OpenShift con il nuovo Certified Operator"
- 27 marzo 2025: "Provisioning Trident per SMB con Google Cloud NetApp Volumes"
- 05 marzo 2025: "Sblocca l'integrazione senza problemi 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: "Introduzione 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 on AWS (ROSA) utilizzando Trident"
- 29 ottobre 2024: "Migrazione live di VM con OpenShift Virtualization su ROSA e Amazon FSx per NetApp ONTAP"
- 08 luglio 2024: "Utilizzo di NVMe/TCP per consumare lo storage ONTAP per le tue moderne applicazioni containerizzate su Amazon EKS"
- 01 luglio 2024: "Storage Kubernetes senza soluzione di continuità con Google Cloud NetApp Volumes Flex e Astra Trident"
- 11 giugno 2024: "ONTAP come storage di backend per il registro delle immagini integrato in OpenShift"

Blog di Trident Protect

- 16 maggio 2025: "Automatizzazione del failover del registro per il disaster recovery con i post-restore hook di Trident Protect"
- 16 maggio 2025: "Disaster recovery della OpenShift Virtualization con NetApp Trident Protect"
- 13 maggio 2025: "Migrazione della storage class con Trident Protect backup restore"
- 09 maggio 2025: "Ridimensiona le applicazioni Kubernetes con i hook post-ripristino di Trident Protect"
- 03 aprile 2025: "Trident Protect Power Up: Replica Kubernetes per la protezione e il disaster recovery"
- 13 marzo 2025: "Operazioni di backup e ripristino coerenti con gli arresti anomali per le VM di OpenShift

Virtualization"

- 11 marzo 2025: "Estensione dei modelli GitOps alla protezione dei dati delle applicazioni con NetApp Trident"
- 03 marzo 2025: "Trident 25.02: Elevare l'esperienza Red Hat OpenShift con nuove entusiasmanti funzionalità"
- 15 gennaio 2025: "Presentazione del controllo degli accessi in base al ruolo di 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 più 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 che vengono rilasciate in una particolare versione di Kubernetes?

Trident solitamente non supporta le funzionalità alpha di Kubernetes. Trident potrebbe supportare le funzionalità beta nelle due release di Trident successive alla release beta di Kubernetes.

Trident ha delle dipendenze da altri prodotti NetApp per il suo funzionamento?

Trident non ha alcuna dipendenza da altri prodotti software NetApp e funziona come applicazione autonoma. Tuttavia, è necessario disporre di un dispositivo di storage NetApp.

Come posso ottenere i dettagli completi della configurazione Trident?

Utilizzare il `tridentctl get` comando per ottenere maggiori informazioni sulla configurazione di Trident.

Posso ottenere metriche su come lo storage viene fornito da Trident?

Sì. Gli endpoint Prometheus possono essere utilizzati per raccogliere informazioni sul funzionamento di Trident, come il numero di backend gestiti, il numero di volumi forniti, i byte consumati e così via. Puoi anche utilizzare "Cloud Insights" per il monitoraggio e l'analisi.

L'esperienza utente cambia quando si utilizza Trident come CSI Provisioner?

No. Non ci sono cambiamenti per quanto riguarda l'esperienza utente e le funzionalità. Il nome del provisioner utilizzato è `csi.trident.netapp.io`. Questo metodo di installazione di Trident è consigliato se si desidera utilizzare tutte le nuove funzionalità fornite dalle versioni attuali e future.

Installa e usa Trident su un cluster Kubernetes

Trident supporta un'installazione offline da un registro privato?

Sì, Trident può essere installato offline. Consulta "Scopri l'installazione di Trident".

Posso installare Trident da remoto?

Sì. Trident 18.10 e versioni successive supportano la funzionalità di installazione remota da qualsiasi macchina che abbia `kubectl` accesso al cluster. Dopo `kubectl` che l'accesso è stato verificato (ad esempio, avviare un `kubectl get nodes` comando dalla macchina remota per la verifica), seguire le istruzioni di installazione.

Posso configurare High Availability con Trident?

Trident viene installato come Kubernetes Deployment (ReplicaSet) con un'istanza, quindi ha HA integrato. Non dovresti aumentare il numero di repliche nella distribuzione. Se il nodo su cui è installato Trident viene perso o il pod è altrimenti inaccessibile, Kubernetes ridistribuisce automaticamente il pod su un nodo funzionante nel tuo cluster. Trident è solo control-plane, quindi i pod attualmente montati non sono influenzati se Trident viene ridistribuito.

Trident ha bisogno di accedere al namespace kube-system?

Trident legge dal Kubernetes API Server per determinare quando le applicazioni richiedono nuovi PVC, quindi ha bisogno di accedere a kube-system.

Quali sono i ruoli e i privilegi 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. Consulta "Personalizza l'installazione di tridentctl".

Posso generare localmente gli esatti file manifest che Trident utilizza per l'installazione?

È possibile generare e modificare localmente i file manifest esatti utilizzati da Trident per l'installazione, se necessario. Consulta "Personalizza l'installazione di tridentctl".

Posso condividere lo stesso backend SVM ONTAP per due istanze Trident separate in due cluster Kubernetes separati?

Sebbene non sia consigliato, è possibile utilizzare la stessa SVM di backend per due istanze Trident. Specificare un nome di volume univoco per ciascuna istanza durante l'installazione e/o specificare un parametro univoco `StoragePrefix` nel file `setup/backend.json`. Questo serve a garantire che lo stesso volume FlexVol non venga utilizzato per entrambe le istanze.

È possibile installare Trident su ContainerLinux (ex CoreOS)?

Trident è semplicemente un pod Kubernetes e può essere installato ovunque Kubernetes sia in esecuzione.

Posso usare Trident con NetApp Cloud Volumes ONTAP?

Sì, Trident è supportato su AWS, Google Cloud e Azure.

Risoluzione dei problemi e supporto

NetApp supporta Trident?

Sebbene Trident sia open source e fornito gratuitamente, NetApp lo supporta pienamente a condizione che il tuo NetApp backend sia supportato.

Come posso aprire un caso di supporto?

Per aprire un caso di supporto, eseguire una delle seguenti operazioni:

1. Contatta il tuo Support Account Manager e ricevi aiuto per aprire un ticket.
2. Invia un caso di supporto contattando "NetApp Supporto".

Come posso generare 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 kubelet per diagnosticare i problemi di montaggio sul lato Kubernetes. Le istruzioni per ottenere il log kubelet variano in base alla modalità di installazione di Kubernetes.

Cosa devo fare se ho bisogno di inoltrare una richiesta per una nuova funzionalità?

Crea un problema su "Trident Github" e menziona **RFE** nell'oggetto e nella descrizione del problema.

Dove posso segnalare un difetto?

Crea un problema su "Trident Github". Assicurati di includere tutte le informazioni e i registri necessari relativi al problema.

Cosa succede se ho una domanda veloce su Trident e ho bisogno di chiarimenti? Esiste una community o un forum?

Se hai domande, problemi o richieste, contattaci tramite il nostro Trident "Canale Discord" o GitHub.

La password del mio sistema storage è cambiata e Trident non funziona più, come posso recuperare?

Aggiorna la password del backend con `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`. Sostituisci `myBackend` nell'esempio con il nome del tuo backend e `` `/path/to_new_backend.json `` con il percorso del file `backend.json` corretto.

Trident non riesce a trovare il mio nodo Kubernetes. Come posso risolvere questo?

Esistono due possibili scenari per cui Trident non riesce a trovare un nodo Kubernetes. Potrebbe trattarsi di un problema di rete all'interno di Kubernetes o di un problema DNS. Il daemonset del nodo Trident che viene 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, questo problema si verifica solo con i nuovi nodi Kubernetes aggiunti al cluster.

Se il pod Trident viene distrutto, perderò i dati?

I dati non andranno persi se il pod Trident viene distrutto. I metadati di Trident sono archiviati negli oggetti CRD. Tutti i PV che sono stati forniti da Trident funzioneranno normalmente.

Aggiorna Trident

Posso effettuare l'aggiornamento direttamente da una versione precedente a una versione più recente (saltando alcune versioni)?

NetApp supporta l'aggiornamento di Trident da una versione principale alla successiva versione principale. È possibile aggiornare dalla versione 18.xx alla 19.xx, dalla 19.xx alla 20.xx e così via. Dovresti testare l'aggiornamento in un laboratorio prima della distribuzione in produzione.

È possibile effettuare il downgrade di Trident a una versione precedente?

Se hai bisogno di una correzione per bug riscontrati dopo un aggiornamento, problemi di dipendenza o un aggiornamento non riuscito o incompleto, dovresti "disinstallare Trident" e reinstallare la versione precedente utilizzando le istruzioni specifiche per quella versione. Questo è l'unico metodo consigliato per effettuare il downgrade a una versione precedente.

Gestisci backend e volumi

Devo definire sia Management che DataLIF in un file di definizione backend ONTAP?

Il LIF di gestione è obbligatorio. Il DataLIF varia:

- ONTAP SAN: Non specificare per iSCSI. Trident utilizza "ONTAP Selective LUN Map" per individuare i LIF iSCSI necessari per stabilire una sessione multipath. Viene generato un avviso se `dataLIF` è definito esplicitamente. Consultare "Opzioni ed esempi di configurazione SAN ONTAP" per i dettagli.
- ONTAP NAS: NetApp consiglia di specificare `dataLIF`. Se non fornito, Trident recupera i dataLIF dall'SVM. Puoi specificare un fully-qualified domain name (FQDN) da usare per le operazioni di mount NFS, permettendoti di creare un DNS round-robin per bilanciare il carico tra più dataLIF. Consultare "Opzioni ed esempi di configurazione NAS ONTAP" per i dettagli

Trident può configurare CHAP per i backend ONTAP?

Sì. Trident supporta la CHAP bidirezionale per i backend ONTAP. Ciò richiede l'impostazione di `useCHAP=true` nella configurazione del backend.

Come si gestiscono le policy di esportazione con Trident?

Trident può creare e gestire dinamicamente le policy di esportazione a partire dalla versione 20.04. Ciò consente all'amministratore dello storage di fornire uno o più blocchi CIDR nella configurazione del backend e di far sì che Trident aggiunga gli IP dei nodi che rientrano in questi intervalli a una policy di esportazione che crea. In questo modo, Trident gestisce automaticamente l'aggiunta e l'eliminazione di regole per i nodi con IP compresi nei CIDR indicati.

È possibile utilizzare indirizzi IPv6 per le Management e DataLIF?

Trident supporta la definizione di indirizzi IPv6 per:

- `managementLIF` e `dataLIF` per ONTAP NAS backends.
- `managementLIF` per ONTAP SAN backends. Non è possibile specificare `dataLIF` su un backend SAN ONTAP.

Trident deve essere installato utilizzando il flag `--use-ipv6` (per `tridentctl` installazione), `IPv6` (per Trident operator), o `tridentTPv6` (per Helm installation) affinché funzioni su IPv6.

È possibile aggiornare il Management LIF sul backend?

Sì, è possibile aggiornare il Management LIF del backend utilizzando il comando `tridentctl update backend`.

È possibile aggiornare il DataLIF sul backend?

È possibile aggiornare il DataLIF su `ontap-nas` e `ontap-nas-economy` solo.

Posso creare più backend in Trident per Kubernetes?

Trident può supportare molti backend contemporaneamente, sia con lo stesso driver che con driver diversi.

In che modo Trident memorizza le credenziali del backend?

Trident memorizza le credenziali del backend come Kubernetes Secrets.

Come fa Trident a selezionare un backend specifico?

Se gli attributi del backend non possono essere utilizzati per selezionare automaticamente i pool giusti per una classe, i parametri `storagePools` e `additionalStoragePools` vengono utilizzati per selezionare un insieme 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 dei pool che Trident utilizza per il provisioning e rimuove tutti i pool che corrispondono.

Se ci sono più backend dello stesso tipo, come fa Trident a selezionare quale backend utilizzare?

Se ci sono più backend configurati dello stesso tipo, Trident seleziona il backend appropriato in base ai parametri presenti in `StorageClass` e `PersistentVolumeClaim`. Ad esempio, se ci sono più backend ontap-nas driver, Trident cerca di abbinare i parametri in `StorageClass` e `PersistentVolumeClaim` combinati e di abbinare un backend che possa soddisfare i requisiti elencati in `StorageClass` e `PersistentVolumeClaim`. Se ci sono più backend che corrispondono alla richiesta, Trident ne seleziona uno a caso.

Trident supporta CHAP bidirezionale con Element/SolidFire?

Sì.

Come distribuisce Trident i Qtrees su un volume ONTAP? Quanti Qtrees possono essere distribuiti su un singolo volume?

Il `ontap-nas-economy` driver crea fino a 200 Qtrees nello stesso volume FlexVol (configurabile tra 50 e 300), 100.000 Qtrees per nodo del cluster e 2,4M per cluster. Quando si inserisce un nuovo `PersistentVolumeClaim` che viene servito dal driver di economia, il driver verifica se esiste già un volume FlexVol in grado di servire il nuovo Qtree. Se non esiste un volume FlexVol in grado di servire il Qtree, viene creato un nuovo volume FlexVol.

Come posso impostare le autorizzazioni Unix per i volumi forniti su ONTAP NAS?

È possibile impostare i permessi Unix sul volume fornito da Trident impostando un parametro nel file di definizione del backend.

Come posso configurare un insieme esplicito di opzioni di montaggio ONTAP NFS 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 Kubernetes Storage Class, seguire l'esempio fornito "qui".

Come posso impostare i volumi approvvigionati su una specifica policy di esportazione?

Per consentire agli host appropriati di accedere a un volume, utilizzare il parametro `exportPolicy` 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 encryption nel file di definizione del backend. Per ulteriori informazioni, fare riferimento a: "Come funziona Trident con NVE e NAE"

Qual è il modo migliore per implementare QoS per ONTAP attraverso Trident?

Usa `StorageClasses` per implementare QoS per ONTAP.

Come si specifica il thin provisioning o il thick provisioning tramite Trident?

I driver ONTAP supportano sia il thin provisioning che il thick provisioning. I driver ONTAP predefiniscono il thin provisioning. Se si desidera 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. Su `StorageClass`, impostare l' `provisioningType` attributo come thick.
2. Nel file di definizione del backend, abilita i volumi spessi impostando `backend spaceReserve parameter` come volume.

Come posso assicurarmi che i volumi utilizzati non vengano eliminati anche se per sbaglio elimino il PVC?

La protezione del PVC viene attivata automaticamente su Kubernetes a partire dalla versione 1.10.

Posso aumentare le dimensioni dei PVC NFS creati da Trident?

Sì. È possibile espandere un PVC che è stato creato da Trident. Si noti che l'autogrow del volume è una funzionalità di ONTAP che non è applicabile a Trident.

Posso importare un volume mentre è in modalità SnapMirror Data Protection (DP) o offline?

L'importazione del volume non riesce se il volume esterno è in modalità DP o è offline. Si riceve 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?

Kubernetes Storage Resource Quota dovrebbe funzionare finché lo storage NetApp ha capacità. Quando lo storage NetApp non può rispettare le impostazioni della quota Kubernetes a causa della mancanza di capacità, Trident tenta di eseguire il provisioning ma restituisce un errore.

Posso creare Snapshot di volume utilizzando Trident?

Sì. La creazione di snapshot di volumi on-demand e di Persistent Volumes da snapshot è supportata da Trident. Per creare PV da snapshot, assicurarsi che la funzione `VolumeSnapshotDataSource` sia stata

abilitata.

Quali sono i driver che supportano le snapshot dei volumi Trident?

Da oggi, il supporto per le istantanee su richiesta è disponibile per i nostri `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san` e `azure-netapp-files` driver backend.

Come posso eseguire un backup snapshot di un volume fornito da Trident con ONTAP?

Questa opzione è disponibile su `ontap-nas`, `ontap-san` e `ontap-nas-flexgroup` driver. È inoltre possibile specificare un `snapshotPolicy` per il `ontap-san-economy` driver al livello FlexVol.

Questa funzione è disponibile anche sui `ontap-nas-economy` driver, ma con una granularità a livello di volume FlexVol e non a livello di qtree. Per abilitare la possibilità di eseguire snapshot dei volumi forniti da Trident, impostare il parametro del backend `snapshotPolicy` sulla policy di snapshot desiderata come definita sul backend ONTAP. Qualsiasi snapshot eseguito dallo storage controller non è conosciuto da Trident.

Posso impostare una percentuale di riserva di snapshot per un volume approvvigionato 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 del backend. Se sono stati configurati `snapshotPolicy` e `snapshotReserve` nel file di definizione del backend, la percentuale di riserva di snapshot viene impostata in base alla percentuale `snapshotReserve` indicata nel file di backend. Se il numero di percentuale `snapshotReserve` non è indicato, ONTAP per impostazione predefinita considera la percentuale di riserva di snapshot come 5. Se l'opzione `snapshotPolicy` è impostata su none, la percentuale di riserva di snapshot è impostata su 0.

Posso accedere direttamente alla directory delle snapshot del volume e copiare i file?

Sì, è possibile accedere alla directory delle snapshot sul volume fornito da Trident impostando il parametro `snapshotDir` nel file di definizione del backend.

Posso configurare SnapMirror per i volumi tramite Trident?

Attualmente, SnapMirror deve essere impostato esternamente utilizzando ONTAP CLI o OnCommand System Manager.

Come posso ripristinare i volumi persistenti a una specifica snapshot di ONTAP?

Per ripristinare un volume a una snapshot di ONTAP, eseguire le seguenti operazioni:

1. Mettere in quiescenza il pod dell'applicazione che sta utilizzando il volume persistente.
2. Ripristina lo snapshot richiesto tramite ONTAP CLI o OnCommand System Manager.
3. Riavvia il pod dell'applicazione.

Trident può effettuare il provisioning dei volumi su SVM che hanno un Load-Sharing Mirror configurato?

I mirror di condivisione del carico possono essere creati per i volumi root delle SVM che servono dati su NFS. ONTAP aggiorna automaticamente i mirror di condivisione del carico per i volumi che sono stati creati da Trident. Ciò può comportare ritardi nel montaggio dei volumi. Quando vengono creati più volumi utilizzando

Trident, il provisioning di un volume dipende dall'aggiornamento del mirror di condivisione del carico da parte di ONTAP.

Come posso separare l'utilizzo della storage class per ogni cliente/tenant?

Kubernetes non consente le classi di storage nei namespace. Tuttavia, puoi utilizzare Kubernetes per limitare l'utilizzo di una specifica classe di storage per namespace utilizzando le Storage Resource Quotas, che sono per namespace. Per negare a uno specifico namespace l'accesso a uno specifico storage, imposta la quota di risorse a 0 per quella classe di storage.

Risoluzione dei problemi

Utilizza i suggerimenti forniti qui per risolvere i problemi che potresti incontrare durante l'installazione e l'utilizzo di Trident.

NOTA	Per ricevere assistenza con Trident, crea un pacchetto di supporto utilizzando `tridentctl logs -a -n trident` e invialo al Supporto NetApp.

Risoluzione dei problemi generali

- Se il Trident pod non si avvia correttamente (ad esempio, quando il Trident pod è bloccato nella `ContainerCreating` fase con meno di due container pronti), eseguire `kubectl -n trident describe deployment trident` e `kubectl -n trident describe pod trident-**--**` può fornire ulteriori informazioni. Ottenere i log di kubelet (ad esempio, tramite `journalctl -xeu kubelet`) può essere utile.
- Se non ci sono informazioni sufficienti nei log di Trident, puoi provare ad abilitare la modalità di debug per Trident passando il `-d` flag al parametro di installazione in base alla tua opzione di installazione.

  Quindi confermare che il debug sia impostato utilizzando `./tridentctl logs -n trident` e cercando `level=debug msg` nel log.

  Installato con Operator

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

  Questo riavvierà tutti i pod Trident, il che potrebbe richiedere diversi secondi. Puoi verificarlo osservando la colonna "AGE" nell'output di `kubectl get pod -n trident`.

  Per Trident 20.07 e 20.10 usare `tprov` in luogo 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 anche possibile ottenere i log di debug per ciascun backend includendo `debugTraceFlags` nella definizione del backend. Ad esempio, includere `debugTraceFlags: {"api":true, "method":true,}` per ottenere le chiamate API e gli attraversamenti dei metodi nei log di Trident. I backend esistenti possono avere `debugTraceFlags` configurato con un `tridentctl backend update`.
- Quando si utilizza Red Hat Enterprise Linux CoreOS (RHCOS), assicurarsi che `iscsid` sia abilitato sui nodi worker e avviato per impostazione predefinita. Questa operazione può essere eseguita utilizzando OpenShift MachineConfigs o modificando i template di ignition.
- 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`sia installato e in esecuzione. Utilizzare il gestore pacchetti richiesto per il sistema operativo host e verificare che `rpcbind`sia in esecuzione. È possibile verificare lo stato del servizio `rpcbind`eseguendo un `systemctl status rpcbind`o un suo equivalente.
- Se un backend Trident segnala che si trova nello stato `failed` nonostante abbia funzionato in precedenza, è probabile che sia causato dalla modifica delle credenziali SVM/admin associate al backend. L'aggiornamento delle informazioni del backend utilizzando `tridentctl update backend` o il riavvio del pod Trident risolverà questo problema.
- Se riscontri problemi di autorizzazione durante l'installazione di Trident con Docker come container runtime, prova a installare Trident con il `--in cluster=false` flag. Questo non utilizzerà un installer pod ed eviterà problemi di autorizzazione riscontrati a causa dell'utente `trident-installer`.
- Utilizzare `uninstall parameter <Uninstalling Trident>` per la pulizia dopo un'esecuzione non riuscita. Per impostazione predefinita, lo script non rimuove i CRD che sono stati creati da Trident, rendendo sicuro disinstallare e installare nuovamente anche in una distribuzione in esecuzione.
- Se vuoi effettuare il downgrade a una versione precedente di Trident, esegui prima il `tridentctl uninstall` comando per rimuovere Trident. Scarica la "Versione Trident" versione desiderata e installala usando il `tridentctl install` comando.
- Dopo un'installazione riuscita, se un PVC è bloccato nella fase `Pending`, l'esecuzione di `kubectl describe pvc` può fornire informazioni aggiuntive sul motivo per cui Trident non è riuscito a fornire un PV per questo PVC.

Distribuzione di Trident non riuscita utilizzando l'operatore

Se si distribuisce Trident tramite l'operatore, lo stato di `TridentOrchestrator` cambia da `Installing` a `Installed`. Se si osserva lo stato `Failed` e l'operatore non è in grado di ripristinarsi autonomamente, è necessario controllare i log dell'operatore eseguendo il seguente comando:

```
tridentctl logs -l trident-operator
```

L'analisi dei log del container trident-operator può indicare dove risiede il problema. Ad esempio, uno di questi problemi potrebbe essere l'impossibilità di estrarre le immagini del container richieste dai registri upstream in

un ambiente airgapped.

Per capire perché l'installazione di Trident non è andata a buon fine, dovresti dare un'occhiata allo `TridentOrchestrator` status.

```
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` che è stato utilizzato per installare Trident. Poiché ogni cluster Kubernetes può avere solo un'istanza di Trident, l'operatore garantisce che in ogni momento esista solo un `TridentOrchestrator` attivo che può creare.

Inoltre, osservare lo stato dei pod Trident può spesso indicare se qualcosa non va.

```
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 vedere chiaramente che i pod non sono in grado di inizializzarsi completamente perché una o più container image non sono state recuperate.

Per risolvere il problema, dovresti modificare il `TridentOrchestrator` CR. In alternativa, puoi eliminare `TridentOrchestrator` e crearne uno nuovo con la definizione modificata e corretta.

Distribuzione di Trident non riuscita utilizzando `tridentctl`

Per capire cosa è andato storto, puoi eseguire nuovamente il programma di installazione utilizzando l'argomento `-d`, che attiverà la modalità debug e ti aiuterà a capire qual è il problema:

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

Dopo aver risolto il problema, puoi pulire l'installazione come segue, e poi eseguire di nuovo il comando `tridentctl install`:

```
./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 CRDs

È possibile rimuovere completamente Trident e tutti i CRD creati e le relative risorse personalizzate associate.

ATTENZIONE	Questa operazione non può essere annullata. Non eseguire questa operazione a meno che non si desideri un'installazione completamente nuova di Trident. Per disinstallare Trident senza rimuovere i CRD, fare riferimento a "Disinstalla 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}}'
```

Helm

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
```

Errore di destaging del nodo NVMe con namespace di blocchi raw RWX su Kubernetes 1.26

Se si utilizza Kubernetes 1.26, l'unstaging dei nodi potrebbe non riuscire quando si utilizza NVMe/TCP con namespace di blocchi raw RWX. I seguenti scenari forniscono una soluzione alternativa al problema. In alternativa, è possibile aggiornare Kubernetes a 1.27.

Eliminato il namespace e il pod

Si consideri uno scenario in cui si dispone di uno spazio dei nomi gestito da Trident (volume persistente NVMe) associato a un pod. Se si elimina lo spazio dei nomi direttamente dal backend ONTAP, il processo di unstaging si blocca dopo il tentativo di eliminare il pod. Questo scenario non influisce sul cluster Kubernetes o su altre funzionalità.

Soluzione alternativa

Smonta il volume persistente (corrispondente a quel namespace) dal rispettivo nodo ed eliminalo.

dataLIF bloccati

```
 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
Avviare i dataLIFS per ripristinare la piena funzionalità.
```

Mappatura dello spazio dei nomi eliminata

```
 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
Aggiungi il  `hostNQN` di nuovo 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 "argomento non valido" quando tentano di montare esportazioni NFSv4.2. Questo problema si verifica quando l' `v4.2-xattrs`opzione non è abilitata sulla SVM. Soluzione alternativa Abilitare l' `v4.2-xattrs`opzione sulla SVM o eseguire l'aggiornamento a ONTAP 9.12.1 o versione successiva, dove questa opzione è abilitata per impostazione predefinita.

Supporto

NetApp fornisce supporto per Trident in diversi modi. Sono disponibili ampie opzioni di auto-assistenza gratuita 24x7, come articoli della knowledgebase (KB) e un canale Discord.

Ciclo di vita del supporto Trident

Trident offre tre livelli di supporto in base alla versione. Consulta "Supporto della versione 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 13-24 dalla data di rilascio.

Auto-supporto
: La documentazione di Trident è disponibile per i mesi 25-36 dalla data di rilascio.

Versione	Supporto completo	Supporto limitato	Auto-supporto

"26,02"	Febbraio 2026	Febbraio 2027	Febbraio 2028
"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

Auto-supporto

Per un elenco completo di articoli sulla risoluzione dei problemi, fare riferimento a "NetApp Knowledgebase (login richiesto)".

Supporto della community

Esiste una vivace community pubblica di utenti di container (inclusi gli sviluppatori Trident) sulla nostra "Canale Discord". Questo è un ottimo posto per porre domande generali sul progetto e discutere di argomenti correlati con colleghi che la pensano allo stesso modo.

NetApp supporto tecnico

Per ricevere assistenza con Trident, crea un pacchetto di supporto utilizzando `tridentctl logs -a -n trident` e invialo a `NetApp Support <Getting Help>`.

Per ulteriori informazioni

- "Risorse Trident"
- "Kubernetes Hub"

Riferimento

Porte Trident

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

Panoramica

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

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

Porte per i nodi controller

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

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

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

Porte per i nodi worker

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

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

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

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

API REST Trident

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

Quando utilizzare l'API REST

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

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

Utilizzo delle REST API

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

L'API funziona come segue:

GET

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

Elenca tutti gli oggetti di quel tipo.

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

Ottiene i dettagli dell'oggetto denominato.

POST

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

Crea un oggetto del tipo specificato.

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

ELIMINA

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

Elimina la risorsa denominata.

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

Opzioni della riga di comando

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

Registrazione

`-debug`

Abilita l'output di debug.

`-loglevel <level>`

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

Kubernetes

`-k8s_pod`

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

-k8s_api_server <insecure-address:insecure-port>

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

Docker

-volume_driver <name>

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

-driver_port <port-number>

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

-config <file>

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

REST

-address <ip-or-host>

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

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

-port <port-number>

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

-rest

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

Oggetti Kubernetes e Trident

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

Come interagiscono gli oggetti tra loro?

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

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

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

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

Oggetti `PersistentVolumeClaim` Kubernetes

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

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

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

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

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

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

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

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

Considera i seguenti punti:

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

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

Oggetti `PersistentVolume` Kubernetes

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

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

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

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

Oggetti `StorageClass` Kubernetes

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

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

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

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

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

I parametri della storage class sono:

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

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

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

Attributi di selezione del pool di storage

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

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

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

[1]: Non supportato dai sistemi ONTAP Select

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

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

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

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

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

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

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

Attributi di Kubernetes

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

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

SUGGERIMENTO

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

  Per i driver di storage NFS, un file system esiste già come parte dell'esportazione NFS. Per utilizzare `fsGroup` la storage class deve comunque specificare un `fsType`. Puoi impostarlo su `nfs` o su qualsiasi valore non nullo.
- Consultare "Espandi volumi" per ulteriori dettagli sull'espansione del volume.
- Il bundle di installazione di Trident fornisce diversi esempi di definizione di storage class da utilizzare con Trident in `sample-input/storage-class-*.yaml`. L'eliminazione di una storage class Kubernetes causa anche l'eliminazione della corrispondente storage class Trident.

Oggetti `VolumeSnapshotClass` Kubernetes

Gli oggetti di Kubernetes `VolumeSnapshotClass` sono analoghi a `StorageClasses`. Aiutano a definire più classi di storage e sono referenziati dalle istantanee di volume per associare l'istantanea alla classe di istantanea richiesta. Ogni istantanea di volume è associata a una singola classe di istantanea di volume.

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

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

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

Oggetti `VolumeSnapshot` Kubernetes

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

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

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

Oggetti `VolumeSnapshotContent` Kubernetes

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

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

nome del PV e del nome dell' `VolumeSnapshotContent`oggetto.

Quando arriva una richiesta di snapshot, Trident crea lo snapshot sul backend. Dopo che lo snapshot è stato creato, Trident configura un `VolumeSnapshotContent` oggetto e quindi espone lo snapshot all'API di Kubernetes.

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

Oggetti `VolumeGroupSnapshotClass` Kubernetes

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

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

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

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

Oggetti `VolumeGroupSnapshot` Kubernetes

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

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

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

Oggetti `VolumeGroupSnapshotContent` Kubernetes

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

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

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

Oggetti `CustomResourceDefinition` Kubernetes

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

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

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

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

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

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

Trident `StorageClass` oggetti

Trident crea classi di storage corrispondenti per oggetti Kubernetes `StorageClass` che specificano

`csi.trident.netapp.io` nel loro campo provisioner. Il nome della classe di storage corrisponde a quello dell'oggetto Kubernetes `StorageClass` che rappresenta.

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

Le classi di archiviazione comprendono una serie di requisiti per i volumi. Trident confronta questi requisiti con gli attributi presenti in ogni pool di storage; se corrispondono, quel pool di storage è un target valido per il provisioning dei volumi che utilizzano quella classe di archiviazione.

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

Oggetti backend Trident

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

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

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

Trident `StoragePool` oggetti

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

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

Trident `Volume` oggetti

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

NOTA

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

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

Attributo	Tipo	Richiesto	Descrizione
versione	stringa	no	Versione dell'API Trident ("1")
nome	stringa	sì	Nome del volume da creare
storageClass	stringa	sì	Classe di storage da utilizzare per il provisioning del volume
dimensione	stringa	sì	Dimensione del volume da fornire in byte
protocollo	stringa	no	Tipo di protocollo da utilizzare; "file" o "blocco"
internalName	stringa	no	Nome dell'oggetto sul sistema storage; generato da Trident
cloneSourceVolume	stringa	no	ontap (nas, san) & solidfire-*: Nome del volume da cui clonare
splitOnClone	stringa	no	ontap (nas, san): Separa il clone dal suo genitore
snapshotPolicy	stringa	no	ontap-*: policy di Snapshot da utilizzare
snapshotReserve	stringa	no	ontap-*: Percentuale del volume riservata alle snapshot
exportPolicy	stringa	no	ontap-nas*: Policy di esportazione da utilizzare
snapshotDirectory	bool	no	ontap-nas*: Se la directory snapshot è visibile
unixPermissions	stringa	no	ontap-nas*: Permessi UNIX iniziali
blockSize	stringa	no	solidfire-*: Dimensione del blocco/settore
fileSystem	stringa	no	Tipo di file system
skipRecoveryQueue	stringa	no	Durante l'eliminazione del volume, ignora la coda di ripristino nello storage ed elimina immediatamente il volume.

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

con i trattini bassi (quindi il nome interno diventa `<prefix>_<volume-name>`). Per i backend Element, sostituisce i trattini bassi con i trattini.

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

Trident `Snapshot` oggetti

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

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

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

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

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

Oggetto Trident `ResourceQuota`

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

A tal fine, Trident impiega un `ResourceQuota` oggetto per garantire che la Priority Class "system-node-

critical" sul daemonset Trident sia soddisfatta. Prima della distribuzione e della creazione del daemonset, Trident cerca il `ResourceQuota` oggetto e, se non viene trovato, lo applica.

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

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

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
    matchExpressions:
      - operator: In
        scopeName: PriorityClass
        values:
          - system-node-critical
```

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

Pulire `ResourceQuota` se l'installazione non riesce

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

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

Rimuovi `ResourceQuota`

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

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

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

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

impostati da Trident.

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

Contesto di sicurezza Kubernetes richiesto e campi correlati

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

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

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

Standard di sicurezza dei Pod (PSS)

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

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

Politiche di sicurezza dei Pod (PSP)

Campo	Descrizione	Predefinito
`allowPrivilegeEscalation`	I container privilegiati devono consentire l'escalation dei privilegi.	`true`
`allowedCSIDrivers`	Trident non utilizza volumi CSI effimeri inline.	Vuoto
`allowedCapabilities`	I container non privilegiati di Trident non richiedono più capacità rispetto al set predefinito e i container privilegiati ricevono tutte le capacità possibili.	Vuoto
`allowedFlexVolumes`	Trident non fa uso di un "Driver FlexVolume", pertanto non sono inclusi nell'elenco dei volumi consentiti.	Vuoto
`allowedHostPaths`	Il pod del nodo Trident monta il filesystem root del nodo, quindi non vi è alcun vantaggio nell'impostare questo elenco.	Vuoto
`allowedProcMountTypes`	Trident non utilizza alcun `ProcMountTypes`.	Vuoto
`allowedUnsafeSysctls`	Trident non richiede alcuna operazione non sicura `sysctls`.	Vuoto
`defaultAddCapabilities`	Non è necessario aggiungere funzionalità ai container privilegiati.	Vuoto
`defaultAllowPrivilegeEscalation`	L'autorizzazione all'escalation dei privilegi viene gestita in ogni pod Trident.	`false`
`forbiddenSysctls`	Non sono consentiti `sysctls`.	Vuoto
`fsGroup`	I container Trident vengono eseguiti come root.	`RunAsAny`

Campo	Descrizione	Predefinito
`hostIPC`	Il montaggio dei volumi NFS richiede che l'host IPC comunichi con `nfsd`	`true`
`hostNetwork`	iscsiadm richiede che la rete host comunichi con il demone iSCSI.	`true`
`hostPID`	È necessario il PID host per verificare se `rpc-statd` è in esecuzione sul nodo.	`true`
`hostPorts`	Trident non utilizza alcuna porta host.	Vuoto
`privileged`	I pod del nodo Trident devono eseguire un container privilegiato per poter montare i volumi.	`true`
`readOnlyRootFilesystem`	I pod del nodo Trident devono scrivere sul file system del nodo.	`false`
`requiredDropCapabilities`	I pod del nodo Trident eseguono un container privilegiato e non possono rilasciare capacità.	`none`
`runAsGroup`	I container Trident vengono eseguiti come root.	`RunAsAny`
`runAsUser`	I container Trident vengono eseguiti come root.	`runAsAny`
`runtimeClass`	Trident non utilizza `RuntimeClasses`.	Vuoto
`seLinux`	Trident non imposta `seLinuxOptions` perché attualmente ci sono differenze nel modo in cui i runtime dei container e le distribuzioni di Kubernetes gestiscono SELinux.	Vuoto
`supplementalGroups`	I container Trident vengono eseguiti come root.	`RunAsAny`
`volumes`	I pod Trident richiedono questi plugin di volume.	`hostPath, projected, emptyDir`

Vincoli del contesto di sicurezza (SCC)

Etichette	Descrizione	Predefinito
`allowHostDirVolumePlugin`	I pod del nodo Trident montano il file system radice del nodo.	`true`
`allowHostIPC`	Il montaggio di volumi NFS richiede che l'host IPC comunichi con `nfsd`.	`true`

Etichette	Descrizione	Predefinito
`allowHostNetwork`	iscsiadm richiede che la rete host comunichi con il demone iSCSI.	`true`
`allowHostPID`	È necessario il PID host per verificare se `rpc-statd` è in esecuzione sul nodo.	`true`
`allowHostPorts`	Trident non utilizza alcuna porta host.	`false`
`allowPrivilegeEscalation`	I container privilegiati devono consentire l'escalation dei privilegi.	`true`
`allowPrivilegedContainer`	I pod del nodo Trident devono eseguire un container privilegiato per poter montare i volumi.	`true`
`allowedUnsafeSysctls`	Trident non richiede alcuna operazione non sicura `sysctls`.	`none`
`allowedCapabilities`	I container non privilegiati di Trident non richiedono più capacità rispetto al set predefinito e i container privilegiati ricevono tutte le capacità possibili.	Vuoto
`defaultAddCapabilities`	Non è necessario aggiungere funzionalità ai container privilegiati.	Vuoto
`fsGroup`	I container Trident vengono eseguiti come root.	`RunAsAny`
`groups`	Questo SCC è specifico per Trident ed è associato al suo utente.	Vuoto
`readOnlyRootFilesystem`	I pod del nodo Trident devono scrivere sul file system del nodo.	`false`
`requiredDropCapabilities`	I pod del nodo Trident eseguono un container privilegiato e non possono rilasciare capacità.	`none`
`runAsUser`	I container Trident vengono eseguiti come root.	`RunAsAny`
`seLinuxContext`	Trident non imposta `seLinuxOptions` perché attualmente ci sono differenze nel modo in cui i runtime dei container e le distribuzioni di Kubernetes gestiscono SELinux.	Vuoto
`seccompProfiles`	I container privilegiati vengono sempre eseguiti come "Unconfined".	Vuoto
`supplementalGroups`	I container Trident vengono eseguiti come root.	`RunAsAny`

Etichette	Descrizione	Predefinito
`users`	Viene fornita una voce per associare questo SCC all'utente Trident nel namespace Trident.	n/d
`volumes`	I pod Trident richiedono questi plugin di volume.	`hostPath, downwardAPI, projected, emptyDir`

Note legali

Le note legali forniscono accesso a dichiarazioni di copyright, marchi, brevetti e altro.

Copyright

"https://www.netapp.com/company/legal/copyright/"

Marchi

NETAPP, il logo NETAPP e i marchi elencati nella pagina dei marchi di NetApp sono marchi di NetApp, Inc. Altri nomi di aziende e prodotti possono 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

Informativa sulla privacy

"https://www.netapp.com/company/legal/privacy-policy/"

Open source

È possibile consultare i diritti d'autore e le licenze di terze parti utilizzati nel NetApp software per Trident nel file delle notifiche per ogni 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.