# Proteggi le applicazioni con Trident Protect

Trident

NetApp
February 02, 2026

# Sommario

# Proteggi le applicazioni con Trident Protect

## Scopri di più su Trident Protect

NetApp Trident Protect offre funzionalità avanzate di gestione dei dati delle applicazioni che migliorano la funzionalità e la disponibilità delle applicazioni Kubernetes con stato supportate dai sistemi di storage NetApp ONTAP e dal provisioner di storage NetApp Trident CSI. Trident Protect semplifica la gestione, la protezione e lo spostamento dei carichi di lavoro containerizzati tra cloud pubblici e ambienti on-premise. Offre inoltre funzionalità di automazione tramite API e CLI.

È possibile proteggere le applicazioni con Trident Protect creando risorse personalizzate (CR) o utilizzando la CLI Trident Protect.

### Quali sono le prossime novità?

Puoi informarti sui requisiti Trident Protect prima di installarlo:

- "Requisiti Trident Protect"

## Installa Trident Protect

### Requisiti Trident Protect

Per iniziare, verifica la prontezza del tuo ambiente operativo, dei cluster applicativi, delle applicazioni e delle licenze. Assicurati che il tuo ambiente soddisfi questi requisiti per distribuire e utilizzare Trident Protect.

#### Compatibilità del cluster Kubernetes Trident Protect

Trident Protect è compatibile con un'ampia gamma di offerte Kubernetes completamente gestite e autogestite, tra cui:

- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- Servizio Kubernetes di Microsoft Azure (AKS)
- Red Hat OpenShift
- SUSE Rancher
- Portfolio VMware Tanzu
- Kubernetes upstream

- I backup Trident Protect sono supportati solo sui nodi di elaborazione Linux. I nodi di elaborazione Windows non sono supportati per le operazioni di backup.
- Assicurarsi che il cluster su cui si installa Trident Protect sia configurato con un controller snapshot in esecuzione e i relativi CRD. Per installare un controller snapshot, fare riferimento a "queste istruzioni" .
- Assicurarsi che esista almeno una VolumeSnapshotClass. Per maggiori informazioni, fare riferimento a"VolumeSnapshotClass" .

**Compatibilità del backend di archiviazione Trident Protect**

Trident Protect supporta i seguenti backend di archiviazione:

- Amazon FSX per NetApp ONTAP
- Cloud Volumes ONTAP
- Array storage ONTAP
- Google Cloud NetApp Volumes
- Azure NetApp Files

Verificare che lo storage backend soddisfi i seguenti requisiti:

- Assicurarsi che lo storage NetApp connesso al cluster utilizzi Trident 24.02 o una versione successiva (si consiglia Trident 24.10).
- Verificare di disporre di un back-end dello storage NetApp ONTAP.
- Verificare di aver configurato un bucket dello storage a oggetti per la memorizzazione dei backup.
- Crea tutti gli spazi dei nomi delle applicazioni che intendi utilizzare per le applicazioni o per le operazioni di gestione dei dati delle applicazioni. Trident Protect non crea questi namespace per te; se specifichi uno namespace inesistente in una risorsa personalizzata, l'operazione non riuscirà.

**Per i volumi nas-Economy**

Trident Protect supporta le operazioni di backup e ripristino sui volumi nas-economy. Snapshot, cloni e replica SnapMirror su volumi nas-economy non sono attualmente supportati. È necessario abilitare una directory snapshot per ogni volume nas-economy che si intende utilizzare con Trident Protect.

Alcune applicazioni non sono compatibili con volumi che utilizzano una directory snapshot. Per queste applicazioni, è necessario nascondere la directory dello snapshot eseguendo il seguente comando nel sistema di archiviazione ONTAP:

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

Puoi abilitare la directory dello snapshot eseguendo il seguente comando per ogni volume di economia nas, sostituendo `<volume-UUID>` con l'UUID del volume che desideri modificare:

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

Per impostazione predefinita, è possibile abilitare le directory snapshot per i nuovi volumi impostando l'opzione di configurazione back-end Trident `snapshotDir` su `true`. I volumi esistenti non vengono influenzati.

### Protezione dei dati con le macchine virtuali KubeVirt

Trident Protect fornisce funzionalità di blocco e sblocco del file system per le macchine virtuali KubeVirt durante le operazioni di protezione dei dati per garantire la coerenza dei dati. Il metodo di configurazione e il comportamento predefinito per le operazioni di blocco delle VM variano a seconda delle versioni Trident Protect, con le versioni più recenti che offrono una configurazione semplificata tramite i parametri del grafico Helm.

Durante le operazioni di ripristino, qualsiasi `VirtualMachineSnapshots` creati per una macchina virtuale (VM) non vengono ripristinati.

#### Trident Protect 25.10 e versioni successive

Trident Protect blocca e sblocca automaticamente i file system KubeVirt durante le operazioni di protezione dei dati per garantire la coerenza. A partire da Trident Protect 25.10, è possibile disattivare questo comportamento utilizzando `vm.freeze` parametro durante l'installazione della carta Helm. Il parametro è abilitato per impostazione predefinita.

```
helm install ... --set vm.freeze=false ...
```

#### Trident Protect dal 24.10.1 al 25.06

A partire da Trident Protect 24.10.1, Trident Protect blocca e sblocca automaticamente i file system KubeVirt durante le operazioni di protezione dei dati. Facoltativamente, è possibile disattivare questo comportamento automatico utilizzando il seguente comando:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=false -n trident-protect
```

**Trident Protect 24.10**

Trident Protect 24.10 non garantisce automaticamente uno stato coerente per i file system delle VM KubeVirt durante le operazioni di protezione dei dati. Se si desidera proteggere i dati della VM KubeVirt utilizzando Trident Protect 24.10, è necessario abilitare manualmente la funzionalità di congelamento/scongelamento per i file system prima dell'operazione di protezione dei dati. Ciò garantisce che i file system siano in uno stato coerente.

È possibile configurare Trident Protect 24.10 per gestire il blocco e lo sblocco del file system della VM durante le operazioni di protezione dei dati tramite"configurazione della virtualizzazione" e quindi utilizzando il seguente comando:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

## Requisiti per la replica SnapMirror

La replica NetApp SnapMirror è disponibile per l'uso con Trident Protect per le seguenti soluzioni ONTAP :

- Cluster NetApp FAS, AFF e ASA on-premise
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSX per NetApp ONTAP

### Requisiti del cluster di ONTAP per la replica SnapMirror

Assicurati che il tuo cluster ONTAP soddisfi i seguenti requisiti se intendi utilizzare la replica SnapMirror:

- * NetApp Trident*: NetApp Trident deve essere presente sia sul cluster Kubernetes di origine che su quello di destinazione che utilizzano ONTAP come backend. Trident Protect supporta la replica con la tecnologia NetApp SnapMirror utilizzando classi di archiviazione supportate dai seguenti driver:
  - `ontap-nas` : NFS
  - `ontap-san` : iSCSI
  - `ontap-san` : FC
  - `ontap-san` : NVMe/TCP (richiede almeno la versione ONTAP 9.15.1)
- **Licenze**: Le licenze asincrone di ONTAP SnapMirror che utilizzano il bundle di protezione dati devono essere attivate sia sul cluster ONTAP di origine che su quello di destinazione. Per ulteriori informazioni, fare riferimento "Panoramica sulle licenze SnapMirror in ONTAP" a.

  A partire da ONTAP 9.10.1, tutte le licenze vengono fornite come file di licenza NetApp (NLF), che è un singolo file che abilita più funzioni. Per ulteriori informazioni, fare riferimento "Licenze incluse con ONTAP ONE" a.

  È supportata solo la protezione asincrona SnapMirror.

#### Considerazioni sul peering per la replica SnapMirror

Assicurati che il tuo ambiente soddisfi i seguenti requisiti se intendi utilizzare il peering di back-end dello storage:

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

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

- **NetApp Trident e SVM**: le SVM remote peered devono essere disponibili per NetApp Trident sul cluster di destinazione.
- **Backend gestiti**: è necessario aggiungere e gestire i backend di archiviazione ONTAP in Trident Protect per creare una relazione di replica.

#### Configurazione Trident / ONTAP per la replica SnapMirror

Trident Protect richiede la configurazione di almeno un backend di archiviazione che supporti la replica sia per i cluster di origine che per quelli di destinazione. Se i cluster di origine e di destinazione sono gli stessi, l'applicazione di destinazione dovrebbe utilizzare un backend di archiviazione diverso da quello dell'applicazione di origine per ottenere la migliore resilienza.

#### Requisiti del cluster Kubernetes per la replica SnapMirror

Assicurati che i tuoi cluster Kubernetes soddisfino i seguenti requisiti:

- **Accessibilità ad AppVault**: sia i cluster di origine che quelli di destinazione devono avere accesso alla rete per leggere e scrivere su AppVault per la replica degli oggetti applicativi.
- **Connettività di rete**: configura le regole del firewall, le autorizzazioni dei bucket e le liste consentite di IP per abilitare la comunicazione tra entrambi i cluster e AppVault attraverso le WAN.

Molti ambienti aziendali implementano rigide policy firewall sulle connessioni WAN. Verificare questi requisiti di rete con il team dell'infrastruttura prima di configurare la replica.

## Installa e configura Trident Protect

Se il tuo ambiente soddisfa i requisiti per Trident Protect, puoi seguire questi passaggi per installare Trident Protect sul tuo cluster. Puoi ottenere Trident Protect da NetApp oppure installarlo dal tuo registro privato. L'installazione da un registro privato è utile se il cluster non riesce ad accedere a Internet.

#### Installa Trident Protect

### Installa Trident Protect da NetApp

**Fasi**

1. Aggiungere il repository Trident Helm:

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

2. Utilizzare Helm per installare Trident Protect. Sostituire `<name-of-cluster>` con un nome cluster, che verrà assegnato al cluster e utilizzato per identificare i backup e gli snapshot del cluster:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2510.0 --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
100.2510.0 --create-namespace --namespace trident-protect
```

La registrazione del debug aiuta NetApp a risolvere i problemi senza dover modificare il livello di registrazione o riprodurre i problemi.

### Installa Trident Protect da un registro privato

È possibile installare Trident Protect da un registro di immagini privato se il cluster Kubernetes non è in grado di accedere a Internet. In questi esempi, sostituisci i valori tra parentesi con le informazioni provenienti dal tuo ambiente:

**Fasi**

1. Estrarre le seguenti immagini sul computer locale, aggiornare i tag e quindi inviarle al registro privato:

```
docker.io/netapp/controller:25.10.0
docker.io/netapp/restic:25.10.0
docker.io/netapp/kopia:25.10.0
docker.io/netapp/kopiablockrestore:25.10.0
docker.io/netapp/trident-autosupport:25.10.0
docker.io/netapp/exechook:25.10.0
docker.io/netapp/resourcebackup:25.10.0
docker.io/netapp/resourcerestore:25.10.0
docker.io/netapp/resourcedelete:25.10.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

Ad esempio:

```
docker pull docker.io/netapp/controller:25.10.0
```

```
docker tag docker.io/netapp/controller:25.10.0 <private-registry-url>/controller:25.10.0
```

```
docker push <private-registry-url>/controller:25.10.0
```

> Per ottenere la tabella Helm, scaricare prima la tabella Helm su un computer con accesso a Internet utilizzando `helm pull trident-protect --version 100.2510.0 --repo https://netapp.github.io/trident-protect-helm-chart`, quindi copia il risultato `trident-protect-100.2510.0.tgz` file nel tuo ambiente offline e installalo utilizzando `helm install trident-protect ./trident-protect-100.2510.0.tgz` invece del riferimento al repository nel passaggio finale.

2. Creare lo spazio dei nomi del sistema Trident Protect:

```
kubectl create ns trident-protect
```

3. Accedere al Registro di sistema:

```
helm registry login <private-registry-url> -u <account-id> -p <api-token>
```

4. Creare un segreto pull da utilizzare per l'autenticazione privata del Registro di sistema:

```
kubectl create secret docker-registry regcred --docker-username=<registry-username> --docker-password=<api-token> -n trident-protect --docker-server=<private-registry-url>
```

5. Aggiungere il repository Trident Helm:

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

6. Crea un file denominato `protectValues.yaml` . Assicurarsi che contenga le seguenti impostazioni Trident Protect:

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

IL `imageRegistry` E `imagePullSecrets` i valori si applicano a tutte le immagini dei componenti, comprese `resourcebackup` E `resourcerestore` . Se si inseriscono immagini in un percorso di repository specifico all'interno del registro (ad esempio, `example.com:443/my-repo`), includere il percorso completo nel campo del registro. Ciò garantirà che tutte le immagini vengano estratte da `<private-registry-url>/<image-name>:<tag>`.

7. Utilizzare Helm per installare Trident Protect. Sostituire `<name_of_cluster>` con un nome cluster, che verrà assegnato al cluster e utilizzato per identificare i backup e gli snapshot del cluster:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name_of_cluster> --version 100.2510.0 --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
100.2510.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

La registrazione del debug aiuta NetApp a risolvere i problemi senza dover modificare il livello di registrazione o riprodurre i problemi.

Per ulteriori opzioni di configurazione del grafico Helm, incluse le impostazioni AutoSupport e il filtraggio dello spazio dei nomi, fare riferimento a "Personalizza l'installazione Trident Protect".

## Installa il plugin Trident Protect CLI

È possibile utilizzare il plugin della riga di comando Trident Protect, che è un'estensione di Trident `tridentctl` utilità, per creare e interagire con le risorse personalizzate (CR) Trident Protect.

### Installa il plugin Trident Protect CLI

Prima di utilizzare l'utilità della riga di comando, è necessario installarla sulla macchina utilizzata per accedere al cluster. Attenersi alla seguente procedura, a seconda che il computer utilizzi una CPU x64 o ARM.

**Scarica il plugin per CPU Linux AMD64**

**Fasi**

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-amd64
```

**Scarica il plugin per CPU Linux ARM64**

**Fasi**

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-arm64
```

**Scarica il plugin per le CPU Mac AMD64**

**Fasi**

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-macos-amd64
```

**Scarica il plugin per le CPU Mac ARM64**

**Fasi**

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-macos-arm64
```

1. Abilitare le autorizzazioni di esecuzione per il binario del plugin:

```
chmod +x tridentctl-protect
```

2. Copiare il file binario del plugin in una posizione definita nella variabile PATH. Ad esempio, `/usr/bin` o `/usr/local/bin` (potrebbe essere necessario un Privileges elevato):

```
cp ./tridentctl-protect /usr/local/bin/
```

3. Facoltativamente, è possibile copiare il file binario del plugin in una posizione nella propria home directory. In questo caso, si consiglia di assicurarsi che la posizione faccia parte della variabile PATH:

```
cp ./tridentctl-protect ~/bin/
```

Copiare il plugin in una posizione nella variabile PATH consente di utilizzare il plugin digitando `tridentctl-protect` o `tridentctl protect` da qualsiasi posizione.

### Visualizza la guida del plugin CLI di Trident

È possibile utilizzare le funzioni della guida del plugin incorporato per ottenere una guida dettagliata sulle funzionalità del plugin:

**Fasi**

1. Utilizzare la funzione di guida per visualizzare le indicazioni sull'utilizzo:

```
tridentctl-protect help
```

### Attivare il completamento automatico del comando

Dopo aver installato il plugin Trident Protect CLI, è possibile abilitare il completamento automatico per determinati comandi.

**Attivare il completamento automatico per la shell Bash**

**Fasi**

1. Creare lo script di completamento:

```
tridentctl-protect completion bash > tridentctl-completion.bash
```

2. Creare una nuova directory nella home directory in modo che contenga lo script:

```
mkdir -p ~/.bash/completions
```

3. Spostare lo script scaricato nella `~/.bash/completions` directory:

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. Aggiungere la seguente riga al `~/.bashrc` file nella propria home directory:

```
source ~/.bash/completions/tridentctl-completion.bash
```

**Attivare il completamento automatico per la shell Z**

**Fasi**

1. Creare lo script di completamento:

```
tridentctl-protect completion zsh > tridentctl-completion.zsh
```

2. Creare una nuova directory nella home directory in modo che contenga lo script:

```
mkdir -p ~/.zsh/completions
```

3. Spostare lo script scaricato nella `~/.zsh/completions` directory:

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. Aggiungere la seguente riga al `~/.zprofile` file nella propria home directory:

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

**Risultato**

Al prossimo login della shell, potete usare il comando auto-completation con il plugin tridentctl-Protect.

## Personalizza l'installazione Trident Protect

È possibile personalizzare la configurazione predefinita di Trident Protect per soddisfare i requisiti specifici del proprio ambiente.

### Specificare i limiti delle risorse del contenitore Trident Protect

Dopo aver installato Trident Protect, è possibile utilizzare un file di configurazione per specificare i limiti delle risorse per i contenitori Trident Protect. Impostando i limiti delle risorse è possibile controllare la quantità di risorse del cluster consumata dalle operazioni Trident Protect.

**Fasi**

1. Creare un file denominato `resourceLimits.yaml`.
2. Compilare il file con le opzioni di limitazione delle risorse per i contenitori Trident Protect in base alle esigenze del proprio ambiente.

   Il seguente file di configurazione di esempio mostra le impostazioni disponibili e contiene i valori predefiniti per ogni limite di risorse:

```
---
jobResources:
  defaults:
    limits:
      cpu: 8000m
      memory: 10000Mi
      ephemeralStorage: ""
    requests:
      cpu: 100m
      memory: 100Mi
      ephemeralStorage: ""
  resticVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  resticVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

```
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

3. Applicare i valori dal `resourceLimits.yaml` file:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

### Personalizzare i vincoli del contesto di protezione

È possibile utilizzare un file di configurazione per modificare i vincoli di contesto di sicurezza (SCC) di OpenShift per i contenitori Trident Protect dopo aver installato Trident Protect. Questi vincoli definiscono le restrizioni di sicurezza per i pod in un cluster Red Hat OpenShift.

**Fasi**

1. Creare un file denominato `sccconfig.yaml`.
2. Aggiungere l'opzione SCC al file e modificare i parametri in base alle esigenze dell'ambiente.

   Nell'esempio seguente vengono mostrati i valori predefiniti dei parametri per l'opzione SCC:

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

Questa tabella descrive i parametri per l'opzione SCC:

| Parametro | Descrizione | Predefinito |
|---|---|---|
| creare | Determina se è possibile creare una risorsa SCC. Una risorsa SCC verrà creata solo se `scc.create` è impostato su `true` e il processo di installazione di Helm identifica un ambiente OpenShift. Se non funziona su OpenShift, o se `scc.create` è impostato su `false`, non verrà creata alcuna risorsa SCC. | vero |
| nome | Specifica il nome della SCC. | processo-di-protezione-Trident |
| priorità | Definisce la priorità dell'SCC. Gli scc con valori di priorità più elevati vengono valutati prima di quelli con valori più bassi. | 1 |

3. Applicare i valori dal `sccconfig.yaml` file:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

In questo modo i valori predefiniti verranno sostituiti con quelli specificati nel `sccconfig.yaml` file.

### Configurare le impostazioni aggiuntive del grafico del timone Trident Protect

È possibile personalizzare le impostazioni AutoSupport e il filtraggio degli spazi dei nomi in base alle proprie esigenze specifiche. La tabella seguente descrive i parametri di configurazione disponibili:

| Parametro | Tipo | Descrizione |
|---|---|---|
| autoSupport.proxy | stringa | Configura un URL proxy per le connessioni NetApp AutoSupport . Utilizzare questa opzione per instradare i caricamenti dei pacchetti di supporto tramite un server proxy. Esempio: `http://my.proxy.url` . |
| autoSupport.insicuro | booleano | Salta la verifica TLS per le connessioni proxy AutoSupport quando impostato su `true` . Utilizzare solo per connessioni proxy non sicure. (predefinito: `false` ) |

| Parametro | Tipo | Descrizione |
| --- | --- | --- |
| autoSupport.abilitato | booleano | Abilita o disabilita i caricamenti giornalieri del bundle Trident Protect AutoSupport . Quando impostato su `false` , i caricamenti giornalieri programmati sono disabilitati, ma puoi comunque generare manualmente i pacchetti di supporto. (predefinito: `true` ) |
| restoreSkipNamespaceAnnotations | stringa | Elenco separato da virgole di annotazioni dello spazio dei nomi da escludere dalle operazioni di backup e ripristino. Consente di filtrare gli spazi dei nomi in base alle annotazioni. |
| ripristina Salta le etichette dello spazio dei nomi | stringa | Elenco separato da virgole delle etichette degli spazi dei nomi da escludere dalle operazioni di backup e ripristino. Consente di filtrare gli spazi dei nomi in base alle etichette. |

È possibile configurare queste opzioni utilizzando un file di configurazione YAML o i flag della riga di comando:

**Utilizzare il file YAML**

**Fasi**

1. Crea un file di configurazione e assegnagli un nome `values.yaml` .
2. Nel file creato, aggiungi le opzioni di configurazione che desideri personalizzare.

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
```

3. Dopo aver popolato il `values.yaml` file con i valori corretti, applicare il file di configurazione:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

**Usa il flag CLI**

**Fasi**

1. Utilizzare il seguente comando con il `--set` flag per specificare parametri individuali:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set-string
restoreSkipNamespaceAnnotations="{annotation1,annotation2}" \
  --set-string restoreSkipNamespaceLabels="{label1,label2}" \
  --reuse-values
```

### Limita i pod Trident Protect a nodi specifici

Puoi utilizzare il vincolo di selezione dei nodi Kubernetes nodeSelector per controllare quali nodi sono idonei a eseguire i pod Trident Protect, in base alle etichette dei nodi. Per impostazione predefinita, Trident Protect è limitato ai nodi che eseguono Linux. È possibile personalizzare ulteriormente questi vincoli in base alle proprie esigenze.

**Fasi**

1. Creare un file denominato `nodeSelectorConfig.yaml`.
2. Aggiungere l'opzione nodeSelector al file e modificare il file per aggiungere o modificare le etichette dei nodi da limitare in base alle esigenze dell'ambiente. Ad esempio, il seguente file contiene la restrizione

predefinita del sistema operativo, ma riguarda anche una regione e un nome dell'applicazione specifici:

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. Applicare i valori dal `nodeSelectorConfig.yaml` file:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

In questo modo, le restrizioni predefinite vengono sostituite da quelle specificate nel `nodeSelectorConfig.yaml` file.

# Gestisci Trident Protect

## Gestisci l'autorizzazione e il controllo degli accessi Trident Protect

Trident Protect utilizza il modello Kubernetes di controllo degli accessi basato sui ruoli (RBAC). Per impostazione predefinita, Trident Protect fornisce un singolo namespace di sistema e il relativo account di servizio predefinito. Se la tua organizzazione ha molti utenti o esigenze di sicurezza specifiche, puoi utilizzare le funzionalità RBAC di Trident Protect per ottenere un controllo più granulare sull'accesso alle risorse e agli spazi dei nomi.

L'amministratore del cluster ha sempre accesso alle risorse nello spazio dei nomi predefinito `trident-protect` e può anche accedere alle risorse in tutti gli altri namespace. Per controllare l'accesso a risorse e applicazioni, è necessario creare spazi dei nomi aggiuntivi e aggiungere risorse e applicazioni a tali spazi dei nomi.

Si noti che nessun utente può creare CRS per la gestione dei dati delle applicazioni nello spazio dei nomi predefinito `trident-protect`. È necessario creare CRS per la gestione dei dati delle applicazioni in uno spazio dei nomi delle applicazioni (come Best practice, creare CRS per la gestione dei dati delle applicazioni nello stesso spazio dei nomi dell'applicazione associata).

Solo gli amministratori dovrebbero avere accesso agli oggetti di risorse personalizzate Trident Protect privilegiati, tra cui:

- **AppVault**: Richiede i dati delle credenziali del bucket
- **AutoSupportBundle**: raccoglie metriche, registri e altri dati sensibili Trident Protect
- **AutoSupportBundleSchedule**: Gestisce i programmi di raccolta dei log

Come Best practice, utilizzare RBAC per limitare l'accesso agli oggetti con privilegi agli amministratori.

Per ulteriori informazioni su come RBAC regola l'accesso alle risorse e agli spazi dei nomi, fare riferimento alla "Documentazione RBAC di Kubernetes" .

Per informazioni sugli account di servizio, fare riferimento alla "Documentazione dell'account del servizio Kubernetes" .

### Esempio: Gestire l'accesso per due gruppi di utenti

Ad esempio, un'organizzazione dispone di un amministratore cluster, di un gruppo di utenti di progettazione e di un gruppo di utenti di marketing. L'amministratore del cluster dovrebbe completare le seguenti attività per creare un ambiente in cui il gruppo di progettazione e il gruppo di marketing hanno ciascuno accesso solo alle risorse assegnate ai rispettivi namespace.

#### Passaggio 1: Creare uno spazio dei nomi che contenga risorse per ciascun gruppo

La creazione di uno spazio dei nomi consente di separare logicamente le risorse e di controllare meglio chi ha accesso a tali risorse.

**Fasi**

1. Creare uno spazio dei nomi per il gruppo tecnico:

```
kubectl create ns engineering-ns
```

2. Creare uno spazio dei nomi per il gruppo di marketing:

```
kubectl create ns marketing-ns
```

#### Passaggio 2: Creare nuovi account di servizio per interagire con le risorse in ogni spazio dei nomi

Ogni nuovo spazio dei nomi creato viene fornito con un account di servizio predefinito, ma è necessario creare un account di servizio per ogni gruppo di utenti in modo da poter dividere ulteriormente Privileges tra i gruppi in futuro, se necessario.

**Fasi**

1. Creare un account di servizio per il gruppo tecnico:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. Creare un account di servizio per il gruppo di marketing:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

#### Passaggio 3: Creare un segreto per ogni nuovo account di servizio

Un segreto dell'account di servizio viene utilizzato per l'autenticazione con l'account di servizio e può essere facilmente eliminato e ricreato se compromesso.

**Fasi**

1. Creare un segreto per l'account del servizio tecnico:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: eng-user
  name: eng-user-secret
  namespace: engineering-ns
type: kubernetes.io/service-account-token
```

2. Creare un segreto per l'account del servizio di marketing:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: mkt-user
  name: mkt-user-secret
  namespace: marketing-ns
type: kubernetes.io/service-account-token
```

#### Passaggio 4: Creare un oggetto RoleBinding per associare l'oggetto ClusterRole a ogni nuovo account di servizio

Quando si installa Trident Protect, viene creato un oggetto ClusterRole predefinito. È possibile associare questo ClusterRole all'account di servizio creando e applicando un oggetto RoleBinding.

**Fasi**

1. Associare ClusterRole all'account del servizio tecnico:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: engineering-ns-tenant-rolebinding
  namespace: engineering-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

2. Associare ClusterRole all'account del servizio di marketing:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: marketing-ns-tenant-rolebinding
  namespace: marketing-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: mkt-user
  namespace: marketing-ns
```

#### Passaggio 5: Verifica delle autorizzazioni

Verificare che le autorizzazioni siano corrette.

**Fasi**

1. Verificare che gli utenti tecnici possano accedere alle risorse di progettazione:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n engineering-ns
```

2. Verificare che gli utenti tecnici non possano accedere alle risorse di marketing:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n marketing-ns
```

#### Passaggio 6: Concedere l'accesso agli oggetti AppVault

Per eseguire attività di gestione dei dati come backup e snapshot, l'amministratore del cluster deve garantire l'accesso agli oggetti AppVault ai singoli utenti.

**Fasi**

1. Creare e applicare un file YAML di combinazione di AppVault e segreto che consenta a un utente di accedere a un AppVault. Ad esempio, la seguente CR concede l'accesso ad AppVault all'utente `eng-user`:

```
apiVersion: v1
data:
  accessKeyID: <ID_value>
  secretAccessKey: <key_value>
kind: Secret
metadata:
  name: appvault-for-eng-user-only-secret
  namespace: trident-protect
type: Opaque
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: appvault-for-eng-user-only
  namespace: trident-protect # Trident Protect system namespace
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: testbucket
      endpoint: 192.168.0.1:30000
      secure: "false"
      skipCertValidation: "true"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: appvault-for-eng-user-only-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: appvault-for-eng-user-only-secret
  providerType: GenericS3
```

2. Creare e applicare un ruolo CR per consentire agli amministratori del cluster di concedere l'accesso a risorse specifiche in uno spazio dei nomi. Ad esempio:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: eng-user-appvault-reader
  namespace: trident-protect
rules:
- apiGroups:
  - protect.trident.netapp.io
  resourceNames:
  - appvault-for-enguser-only
  resources:
  - appvaults
  verbs:
  - get
```

3. Creare e applicare un RoleBinding CR per associare le autorizzazioni all'utente eng-user. Ad esempio:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eng-user-read-appvault-binding
  namespace: trident-protect
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: eng-user-appvault-reader
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

4. Verificare che le autorizzazioni siano corrette.

   a. Tentativo di recuperare le informazioni sull'oggetto AppVault per tutti gli spazi dei nomi:

   ```
   kubectl get appvaults -n trident-protect
   --as=system:serviceaccount:engineering-ns:eng-user
   ```

   L'output dovrebbe essere simile a quanto segue:

```
Error from server (Forbidden): appvaults.protect.trident.netapp.io is
forbidden: User "system:serviceaccount:engineering-ns:eng-user"
cannot list resource "appvaults" in API group
"protect.trident.netapp.io" in the namespace "trident-protect"
```

b. Verificare se l'utente può ottenere le informazioni AppVault a cui ora dispone dell'autorizzazione per accedere:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get appvaults.protect.trident.netapp.io/appvault-for-eng-user-only -n
trident-protect
```

L'output dovrebbe essere simile a quanto segue:

```
yes
```

**Risultato**

Gli utenti a cui sono state concesse le autorizzazioni AppVault dovrebbero essere in grado di utilizzare gli oggetti AppVault autorizzati per le operazioni di gestione dei dati delle applicazioni e non dovrebbero essere in grado di accedere a risorse esterne agli spazi dei nomi assegnati o creare nuove risorse a cui non hanno accesso.

## Monitorare le risorse Trident Protect

È possibile utilizzare gli strumenti open source kube-state-metrics, Prometheus e Alertmanager per monitorare lo stato di integrità delle risorse protette da Trident Protect.

Il servizio kube-state-metrics genera metriche dalla comunicazione API di Kubernetes. Utilizzandolo con Trident Protect, puoi ottenere informazioni utili sullo stato delle risorse nel tuo ambiente.

Prometheus è un toolkit in grado di acquisire i dati generati da kube-state-metrics e presentarli come informazioni facilmente leggibili su questi oggetti. Insieme, kube-state-metrics e Prometheus ti consentono di monitorare lo stato e l'integrità delle risorse che gestisci con Trident Protect.

Alertmanager è un servizio che acquisisce gli avvisi inviati da strumenti come Prometheus e li indirizza alle destinazioni configurate dall'utente.

Le configurazioni e le istruzioni incluse in questa procedura sono solo esempi; è necessario personalizzarle in base all'ambiente in uso. Per istruzioni specifiche e assistenza, consultare la seguente documentazione ufficiale:

- "documentazione kube-state-metrics"
- "Documentazione Prometheus"
- "Documentazione di Alertmanager"

### Fase 1: Installare gli strumenti di monitoraggio

Per abilitare il monitoraggio delle risorse in Trident Protect, è necessario installare e configurare kube-state-metrics, Prometheus e Alertmanager.

#### Installa metriche-stato-kube

È possibile installare parametri kube-state-metrics utilizzando Helm.

**Fasi**

1. Aggiungere il grafico Helm kube-state-metrics. Ad esempio:

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update
```

2. Applicare il CRD di Prometheus ServiceMonitor al cluster:

```
kubectl apply -f https://raw.githubusercontent.com/prometheus-
operator/prometheus-operator/main/example/prometheus-operator-
crd/monitoring.coreos.com_servicemonitors.yaml
```

3. Creare un file di configurazione per il grafico Helm (ad esempio, `metrics-config.yaml`). È possibile personalizzare la seguente configurazione di esempio in base all'ambiente in uso:

**Metrics-config.yaml: Configurazione del grafico Helm kube-state-metrics**

```
---
extraArgs:
  # Collect only custom metrics
  - --custom-resource-state-only=true

customResourceState:
  enabled: true
  config:
    kind: CustomResourceStateMetrics
    spec:
      resources:
      - groupVersionKind:
          group: protect.trident.netapp.io
          kind: "Backup"
          version: "v1"
        labelsFromPath:
          backup_uid: [metadata, uid]
          backup_name: [metadata, name]
          creation_time: [metadata, creationTimestamp]
        metrics:
        - name: backup_info
          help: "Exposes details about the Backup state"
          each:
            type: Info
            info:
              labelsFromPath:
                appVaultReference: ["spec", "appVaultRef"]
                appReference: ["spec", "applicationRef"]
rbac:
  extraRules:
  - apiGroups: ["protect.trident.netapp.io"]
    resources: ["backups"]
    verbs: ["list", "watch"]

# Collect metrics from all namespaces
namespaces: ""

# Ensure that the metrics are collected by Prometheus
prometheus:
  monitor:
    enabled: true
```

4. Installare le metriche di stato kube distribuendo il grafico Helm. Ad esempio:

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. Configurare kube-state-metrics per generare metriche per le risorse personalizzate utilizzate da Trident Protect seguendo le istruzioni in "documentazione sulle risorse personalizzate kube-state-metrics" .

#### Installare Prometheus

È possibile installare Prometheus seguendo le istruzioni riportate nella "Documentazione Prometheus" .

#### Installare Alertmanager

È possibile installare Alertmanager seguendo le istruzioni riportate nella "Documentazione di Alertmanager" .

### Fase 2: Configurare gli strumenti di monitoraggio per lavorare insieme

Dopo aver installato gli strumenti di monitoraggio, è necessario configurarli per lavorare insieme.

**Fasi**

1. Integra metriche-stato-kube con Prometheus. Modificare il file di configurazione di Prometheus (`prometheus.yaml`) e aggiungere le informazioni del servizio kube-state-metrics. Ad esempio:

   **prometheus.yaml: integrazione del servizio kube-state-metrics con Prometheus**

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: trident-protect
data:
  prometheus.yaml: |
    global:
      scrape_interval: 15s
    scrape_configs:
      - job_name: 'kube-state-metrics'
        static_configs:
          - targets: ['kube-state-metrics.trident-protect.svc:8080']
```

2. Configurare Prometheus per instradare gli avvisi ad Alertmanager. Modificare il file di configurazione di Prometheus (`prometheus.yaml`) e aggiungere la seguente sezione:

**prometheus.yaml: Invia avvisi ad Alertmanager**

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager.trident-protect.svc:9093
```

**Risultato**

Prometheus può ora raccogliere le metriche dalle metriche dello stato del kube e inviare avvisi ad Alertmanager. Ora si è pronti a configurare quali condizioni attivano un avviso e dove inviare gli avvisi.

### Passaggio 3: Configurare le destinazioni degli avvisi e degli avvisi

Dopo aver configurato gli strumenti per lavorare insieme, è necessario configurare il tipo di informazioni che attivano gli avvisi e la posizione in cui devono essere inviati.

#### Esempio di avviso: Errore di backup

Nell'esempio seguente viene definito un avviso critico che viene attivato quando lo stato della risorsa personalizzata di backup è impostato su `Error` per 5 secondi o più. È possibile personalizzare questo esempio in base all'ambiente in uso e includere questo frammento YAML nel `prometheus.yaml` file di configurazione:

**rules.yaml: Definisci un avviso Prometheus per i backup non riusciti**

```
rules.yaml: |
  groups:
    - name: fail-backup
        rules:
          - alert: BackupFailed
            expr: kube_customresource_backup_info{status="Error"}
            for: 5s
            labels:
              severity: critical
            annotations:
              summary: "Backup failed"
              description: "A backup has failed."
```

#### Configurare Alertmanager per inviare avvisi ad altri canali

È possibile configurare Alertmanager in modo che invii notifiche ad altri canali, quali e-mail, PagerDuty, Microsoft Teams o altri servizi di notifica specificando la rispettiva configurazione nel `alertmanager.yaml` file.

Nell'esempio seguente, Alertmanager configura l'invio di notifiche a un canale Slack. Per personalizzare questo esempio in base all'ambiente in uso, sostituire il valore della `api_url` chiave con l'URL slack webhook utilizzato nell'ambiente in uso:

**alertmanager.yaml: invia avvisi a un canale Slack**

```
data:
  alertmanager.yaml: |
    global:
      resolve_timeout: 5m
    route:
      receiver: 'slack-notifications'
    receivers:
      - name: 'slack-notifications'
        slack_configs:
          - api_url: '<your-slack-webhook-url>'
            channel: '#failed-backups-channel'
            send_resolved: false
```

## Genera un pacchetto di supporto Trident Protect

Trident Protect consente agli amministratori di generare bundle che includono informazioni utili al supporto NetApp , tra cui registri, metriche e informazioni sulla topologia dei cluster e delle app in gestione. Se sei connesso a Internet, puoi caricare i bundle di supporto sul sito di supporto NetApp (NSS) utilizzando un file di risorse personalizzato (CR).

**Creare un pacchetto di supporto utilizzando una CR**

**Fasi**

1. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-support-bundle.yaml`).
2. Configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.triggerType**: (*required*) determina se il bundle di supporto viene generato immediatamente o pianificato. La generazione pianificata del pacchetto avviene alle 12am:00 UTC. Valori possibili:
     - Pianificato
     - Manuale
   - **Spec.uploadEnabled**: (*Optional*) Controlla se il bundle di supporto deve essere caricato nel sito di supporto NetApp dopo che è stato generato. Se non specificato, il valore predefinito è `false` . Valori possibili:
     - vero
     - false (impostazione predefinita)
   - **Spec.dataWindowStart**: (*Optional*) stringa di data in formato RFC 3339 che specifica la data e l'ora di inizio della finestra dei dati inclusi nel pacchetto di supporto. Se non specificato, il valore predefinito è 24 ore fa. La prima data della finestra che è possibile specificare è 7 giorni fa.

     Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AutoSupportBundle
metadata:
  name: trident-protect-support-bundle
spec:
  triggerType: Manual
  uploadEnabled: true
  dataWindowStart: 2024-05-05T12:30:00Z
```

3. Dopo aver popolato il `trident-protect-support-bundle.yaml` file con i valori corretti, applicare CR:

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-protect
```

**Creare un bundle di supporto utilizzando la CLI**

**Fasi**

1. Creare il pacchetto di supporto, sostituendo i valori tra parentesi con le informazioni dell'ambiente. `trigger-type`Determina se il bundle viene creato immediatamente o se l'ora

di creazione è dettata dalla pianificazione e può essere `Manual` o `Scheduled`. L'impostazione predefinita è `Manual`.

Ad esempio:

```
tridentctl-protect create autosupportbundle <my-bundle-name>
--trigger-type <trigger-type> -n trident-protect
```

**Monitorare e recuperare il pacchetto di supporto**

Dopo aver creato un pacchetto di supporto utilizzando uno dei due metodi, puoi monitorarne l'avanzamento della generazione e recuperarlo nel tuo sistema locale.

**Fasi**

1. Aspetta il `status.generationState` raggiungere `Completed` stato. È possibile monitorare l'avanzamento della generazione con il seguente comando:

   ```
   kubectl get autosupportbundle trident-protect-support-bundle -n trident-protect
   ```

2. Recupera il pacchetto di supporto sul tuo sistema locale. Ottieni il comando di copia dal bundle AutoSupport completato:

   ```
   kubectl describe autosupportbundle trident-protect-support-bundle -n trident-protect
   ```

   Trova il `kubectl cp` comando dall'output ed eseguilo, sostituendo l'argomento di destinazione con la directory locale preferita.

## Aggiorna Trident Protect

Puoi aggiornare Trident Protect all'ultima versione per sfruttare le nuove funzionalità o le correzioni di bug.

- Quando si esegue l'aggiornamento dalla versione 24.10, gli snapshot in esecuzione durante l'aggiornamento potrebbero non funzionare. Questo problema non impedisce la creazione di snapshot futuri, manuali o pianificati. Se uno snapshot non funziona durante l'aggiornamento, è possibile crearne manualmente uno nuovo per garantire la protezione dell'applicazione.

  Per evitare potenziali errori, è possibile disabilitare tutte le pianificazioni degli snapshot prima dell'aggiornamento e riabilitarle in seguito. Tuttavia, ciò comporterà la perdita di tutti gli snapshot pianificati durante il periodo di aggiornamento.

- Per le installazioni di registri privati, assicurati che il grafico Helm e le immagini richiesti per la versione di destinazione siano disponibili nel tuo registro privato e verifica che i tuoi valori Helm personalizzati siano compatibili con la nuova versione del grafico. Per maggiori informazioni, fare riferimento a"Installa Trident Protect da un registro privato" .

Per aggiornare Trident Protect, procedere come segue.

**Fasi**

1. Aggiornare il repository di Trident Helm:

```
helm repo update
```

2. Aggiorna i CRD Trident Protect:

   Questo passaggio è necessario se si esegue l'aggiornamento da una versione precedente alla 25.06, poiché i CRD sono ora inclusi nella tabella Trident Protect Helm.

   a. Eseguire questo comando per spostare la gestione dei CRD da `trident-protect-crds` A `trident-protect` :

```
kubectl get crd | grep protect.trident.netapp.io | awk '{print $1}' |
xargs -I {} kubectl patch crd {} --type merge -p '{"metadata":
{"annotations":{"meta.helm.sh/release-name": "trident-protect"}}}'
```

   b. Esegui questo comando per eliminare il segreto Helm per `trident-protect-crds` grafico:

   Non disinstallare il `trident-protect-crds` grafico utilizzando Helm, poiché ciò potrebbe rimuovere i CRD e tutti i dati correlati.

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. Aggiorna Trident Protect:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2510.0 --namespace trident-protect
```

È possibile configurare il livello di registrazione durante l'aggiornamento aggiungendo `--set logLevel=debug` al comando di aggiornamento. Il livello di registrazione predefinito è `warn` . La registrazione del debug è consigliata per la risoluzione dei problemi, poiché aiuta il supporto NetApp a diagnosticare i problemi senza richiedere modifiche al livello di registro o la riproduzione del problema.

# Gestisci e proteggi le applicazioni

## Utilizzare gli oggetti Trident Protect AppVault per gestire i bucket

La risorsa personalizzata del bucket (CR) per Trident Protect è nota come AppVault. Gli oggetti AppVault sono la rappresentazione dichiarativa del flusso di lavoro Kubernetes di un bucket di archiviazione. Un CR di AppVault contiene le configurazioni necessarie affinché un bucket venga utilizzato nelle operazioni di protezione, come backup, snapshot, operazioni di ripristino e replica SnapMirror . Solo gli amministratori possono creare AppVault.

Quando si eseguono operazioni di protezione dei dati su un'applicazione, è necessario creare una CR di AppVault manualmente o dalla riga di comando. La CR di AppVault è specifica per il proprio ambiente e gli esempi in questa pagina possono essere utilizzati come guida per la creazione di CR di AppVault.

Assicurarsi che AppVault CR si trovi nel cluster in cui è installato Trident Protect. Se la CR di AppVault non esiste o non è possibile accedervi, la riga di comando mostrerà un errore.

### Configurare l'autenticazione e le password AppVault

Prima di creare una CR di AppVault, assicurati che AppVault e il data mover scelto possano autenticarsi con il provider e con tutte le risorse correlate.

#### Password del repository di spostamento dati

Quando si creano oggetti AppVault utilizzando CR o il plug-in Trident Protect CLI, è possibile specificare un segreto Kubernetes con password personalizzate per la crittografia Restic e Kopia. Se non specifichi un segreto, Trident Protect utilizza una password predefinita.

- Quando si creano manualmente CR di AppVault, utilizzare il campo **spec.dataMoverPasswordSecretRef** per specificare il segreto.
- Quando si creano oggetti AppVault utilizzando la CLI Trident Protect, utilizzare `--data-mover -password-secret-ref` argomento per specificare il segreto.

### Creare una password segreta dell'archivio di spostamento dati

Utilizzare i seguenti esempi per creare la password segreta. Quando si creano oggetti AppVault, è possibile indicare a Trident Protect di utilizzare questo segreto per l'autenticazione con il repository del data mover.

- A seconda di quale strumento di spostamento dati si sta utilizzando, è sufficiente includere la password corrispondente per tale strumento. Ad esempio, se si sta utilizzando Restic e non si prevede di utilizzare Kopia in futuro, è possibile includere solo la password Restic quando si crea il segreto.
- Conservare la password in un luogo sicuro. Sarà necessaria per ripristinare i dati sullo stesso cluster o su uno diverso. Se il cluster o il `trident-protect` Se lo spazio dei nomi viene eliminato, non sarà possibile ripristinare i backup o gli snapshot senza la password.

**Utilizzare un CR**

```
---
apiVersion: v1
data:
  KOPIA_PASSWORD: <base64-encoded-password>
  RESTIC_PASSWORD: <base64-encoded-password>
kind: Secret
metadata:
  name: my-optional-data-mover-secret
  namespace: trident-protect
type: Opaque
```

**Utilizzare la CLI**

```
kubectl create secret generic my-optional-data-mover-secret \
--from-literal=KOPIA_PASSWORD=<plain-text-password> \
--from-literal=RESTIC_PASSWORD=<plain-text-password> \
-n trident-protect
```

#### Autorizzazioni IAM per l'archiviazione compatibile con S3

Quando si accede a un archivio compatibile con S3 come Amazon S3, Generic S3, "StorageGRID S3" , O "ONTAP S3" utilizzando Trident Protect, è necessario assicurarsi che le credenziali utente fornite dispongano delle autorizzazioni necessarie per accedere al bucket. Di seguito è riportato un esempio di policy che concede le autorizzazioni minime richieste per l'accesso con Trident Protect. È possibile applicare questo criterio all'utente che gestisce i criteri dei bucket compatibili con S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

Per ulteriori informazioni sulle policy di Amazon S3, fare riferimento agli esempi in "Documentazione di Amazon S3" .

#### Identità pod EKS per l'autenticazione Amazon S3 (AWS)

Trident Protect supporta EKS Pod Identity per le operazioni di spostamento dati Kopia. Questa funzionalità consente l'accesso sicuro ai bucket S3 senza dover archiviare le credenziali AWS nei segreti di Kubernetes.

**Requisiti per l'identità del pod EKS con Trident Protect**

Prima di utilizzare EKS Pod Identity con Trident Protect, accertarsi di quanto segue:

- Il tuo cluster EKS ha l'identità Pod abilitata.
- Hai creato un ruolo IAM con le autorizzazioni necessarie per il bucket S3. Per saperne di più, fare riferimento a"Autorizzazioni IAM per l'archiviazione compatibile con S3" .
- Il ruolo IAM è associato ai seguenti account di servizio Trident Protect:
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

Per istruzioni dettagliate sull'abilitazione di Pod Identity e sull'associazione dei ruoli IAM agli account di servizio, fare riferimento a "Documentazione sull'identità del pod AWS EKS" .

**Configurazione AppVault** Quando si utilizza EKS Pod Identity, configurare il CR AppVault con `useIAM: true` flag invece di credenziali esplicite:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: eks-protect-vault
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-aws
      endpoint: s3.example.com
      useIAM: true
```

#### Esempi di generazione delle chiavi AppVault per i cloud provider

Quando si definisce un CR AppVault, è necessario includere le credenziali per accedere alle risorse ospitate dal provider, a meno che non si utilizzi l'autenticazione IAM. Il modo in cui vengono generate le chiavi per le credenziali varia a seconda del provider. Di seguito sono riportati esempi di generazione di chiavi da riga di comando per diversi provider. È possibile utilizzare gli esempi seguenti per creare chiavi per le credenziali di ciascun provider cloud.

**Google Cloud**

```
kubectl create secret generic <secret-name> \
--from-file=credentials=<mycreds-file.json> \
-n trident-protect
```

**Amazon S3 (AWS)**

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<amazon-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

**Microsoft Azure**

```
kubectl create secret generic <secret-name> \
--from-literal=accountKey=<secret-name> \
-n trident-protect
```

**Generico S3**

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<generic-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

**ONTAP S3**

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<ontap-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

**StorageGRID S3**

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<storagegrid-s3-trident-protect-src
-bucket-secret> \
-n trident-protect
```

### Esempi di creazione di AppVault

Di seguito sono riportate alcune definizioni AppVault di esempio per ogni provider.

#### Esempi di AppVault CR

È possibile utilizzare i seguenti esempi CR per creare oggetti AppVault per ciascun provider cloud.

- Puoi anche specificare un Kubernetes Secret che contiene password personalizzate per la crittografia dei repository Restic e Kopia. Per ulteriori informazioni, fare riferimento Password del repository di spostamento dati a.
- Per gli oggetti AppVault di Amazon S3 (AWS), è possibile specificare un oggetto sessionToken, utile se si utilizza il Single Sign-on (SSO) per l'autenticazione. Questo token viene creato quando si generano le chiavi per il provider in Esempi di generazione delle chiavi AppVault per i cloud provider.
- Per gli oggetti AppVault S3, è possibile specificare facoltativamente un URL proxy di uscita per il traffico S3 in uscita utilizzando la `spec.providerConfig.S3.proxyURL` chiave.

**Google Cloud**

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

**Amazon S3 (AWS)**

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: amazon-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
    sessionToken:
      valueFromSecret:
        key: sessionToken
        name: s3-secret
```

Per gli ambienti EKS che utilizzano Pod Identity con Kopia Data Mover, è possibile rimuovere `providerCredentials` sezione e aggiungi `useIAM: true` sotto il `s3` configurazione invece.

**Microsoft Azure**

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

**Generico S3**

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: generic-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

**ONTAP S3**

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: OntapS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

**StorageGRID S3**

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

#### Esempi di creazione di AppVault utilizzando la CLI Trident Protect

È possibile utilizzare i seguenti esempi di comandi CLI per creare CRS AppVault per ciascun provider.

- Puoi anche specificare un Kubernetes Secret che contiene password personalizzate per la crittografia dei repository Restic e Kopia. Per ulteriori informazioni, fare riferimento Password del repository di spostamento dati a.
- Per gli oggetti AppVault S3, è possibile specificare facoltativamente un URL proxy di uscita per il traffico S3 in uscita utilizzando l' `--proxy-url <ip_address:port>`argomento.

**Google Cloud**

```
tridentctl-protect create vault GCP <vault-name> \
--bucket <mybucket> \
--project <my-gcp-project> \
--secret <secret-name>/credentials \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

**Amazon S3 (AWS)**

```
tridentctl-protect create vault AWS <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

**Microsoft Azure**

```
tridentctl-protect create vault Azure <vault-name> \
--account <account-name> \
--bucket <bucket-name> \
--secret <secret-name> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

**Generico S3**

```
tridentctl-protect create vault GenericS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

**ONTAP S3**

```
tridentctl-protect create vault OntapS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

**StorageGRID S3**

```
tridentctl-protect create vault StorageGridS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

#### Supportato `providerConfig.s3` opzioni di configurazione

Per le opzioni di configurazione del provider S3, consultare la seguente tabella:

| Parametro | Descrizione | Predefinito | Esempio |
|---|---|---|---|
| `providerConfig.s3.skipCertValidation` | Disattiva la verifica del certificato SSL/TLS. | falso | "vero", "falso" |
| `providerConfig.s3.secure` | Abilita la comunicazione HTTPS sicura con l'endpoint S3. | vero | "vero", "falso" |
| `providerConfig.s3.proxyURL` | Specificare l'URL del server proxy utilizzato per connettersi a S3. | Nessuno | http://proxy.example.com:8080 |
| `providerConfig.s3.rootCA` | Fornire un certificato CA radice personalizzato per la verifica SSL/TLS. | Nessuno | "CN=MyCustomCA" |
| `providerConfig.s3.useIAM` | Abilita l'autenticazione IAM per accedere ai bucket S3. Applicabile per l'identità del pod EKS. | falso | vero, falso |

### Visualizzare le informazioni AppVault

È possibile utilizzare il plug-in Trident Protect CLI per visualizzare informazioni sugli oggetti AppVault creati nel cluster.

**Fasi**

1. Visualizzare il contenuto di un oggetto AppVault:

```
tridentctl-protect get appvaultcontent gcp-vault \
--show-resources all \
-n trident-protect
```

**Output di esempio**:

```
+-------------+-------+----------+----------------------------
+---------------------------+
|   CLUSTER   |  APP  |   TYPE   |            NAME             |
TIMESTAMP         |
+-------------+-------+----------+----------------------------
+---------------------------+
|             | mysql | snapshot | mysnap                      | 2024-
08-09 21:02:11 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815180300 | 2024-
08-15 18:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815200300 | 2024-
08-15 20:03:06 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815180300 | 2024-
08-15 18:04:25 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:30 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815200300 | 2024-
08-15 20:04:21 (UTC) |
| production1 | mysql | backup   | mybackup5                   | 2024-
08-09 22:25:13 (UTC) |
|             | mysql | backup   | mybackup                    | 2024-
08-09 21:02:52 (UTC) |
+-------------+-------+----------+----------------------------
+---------------------------+
```

2. Facoltativamente, per visualizzare AppVaultPath per ogni risorsa, utilizzare il flag `--show-paths`.

   Il nome del cluster nella prima colonna della tabella è disponibile solo se è stato specificato un nome del cluster nell'installazione di Trident Protect Helm. Per esempio: `--set clusterName=production1` .

### Rimuovere un AppVault

È possibile rimuovere un oggetto AppVault in qualsiasi momento.

Non rimuovere la `finalizers` chiave in AppVault CR prima di eliminare l'oggetto AppVault. In tal caso, i dati residui nel bucket AppVault e le risorse orfane nel cluster possono risultare.

**Prima di iniziare**

Assicurarsi di aver eliminato tutti i CRS di backup e snapshot utilizzati dall'AppVault che si desidera eliminare.

**Rimuovere un AppVault usando l'interfaccia a riga di comando di Kubernetes**

1. Rimuovere l'oggetto AppVault, sostituendo `appvault-name` con il nome dell'oggetto AppVault da rimuovere:

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

**Rimuovere un AppVault utilizzando la CLI Trident Protect**

1. Rimuovere l'oggetto AppVault, sostituendo `appvault-name` con il nome dell'oggetto AppVault da rimuovere:

```
tridentctl-protect delete appvault <appvault-name> \
-n trident-protect
```

## Definisci un'applicazione per la gestione con Trident Protect

È possibile definire un'applicazione che si desidera gestire con Trident Protect creando una CR dell'applicazione e una CR AppVault associata.

### Creare un AppVault CR

È necessario creare un CR AppVault che verrà utilizzato durante l'esecuzione di operazioni di protezione dei dati sull'applicazione e il CR AppVault deve risiedere nel cluster in cui è installato Trident Protect. Il CR di AppVault è specifico per il tuo ambiente; per esempi di CR di AppVault, fai riferimento a"Risorse personalizzate AppVault."

### Definire un'applicazione

È necessario definire ciascuna applicazione che si desidera gestire con Trident Protect. È possibile definire un'applicazione per la gestione creando manualmente un CR dell'applicazione oppure utilizzando la CLI Trident Protect.

**Aggiungere un'applicazione utilizzando una CR**

**Fasi**

1. Creare il file CR dell'applicazione di destinazione:
   a. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `maria-app.yaml`).
   b. Configurare i seguenti attributi:
      - **metadata.name**: (*required*) il nome della risorsa personalizzata dell'applicazione. Si noti il nome scelto perché altri file CR necessari per le operazioni di protezione fanno riferimento a questo valore.
      - **spec.includedNamespaces**: (*required*) utilizzare lo spazio dei nomi e il selettore di etichette per specificare gli spazi dei nomi e le risorse utilizzate dall'applicazione. Lo spazio dei nomi dell'applicazione deve far parte di questo elenco. Il selettore delle etichette è opzionale e può essere utilizzato per filtrare le risorse all'interno di ogni spazio dei nomi specificato.
      - **spec.includedClusterScopedResources**: (*Optional*) utilizzare questo attributo per specificare le risorse con ambito cluster da includere nella definizione dell'applicazione. Questo attributo consente di selezionare queste risorse in base al gruppo, alla versione, al tipo e alle etichette.
        - **GroupVersionKind**: (*required*) specifica il gruppo API, la versione e il tipo di risorsa con ambito cluster.
        - **LabelSelector**: (*Optional*) Filtra le risorse con ambito cluster in base alle loro etichette.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Facoltativo*) Questa annotazione è applicabile solo alle applicazioni definite da macchine virtuali, come negli ambienti KubeVirt, in cui i blocchi del file system si verificano prima degli snapshot. Specificare se questa applicazione può scrivere sul file system durante uno snapshot. Se impostato su true, l'applicazione ignora l'impostazione globale e può scrivere sul file system durante uno snapshot. Se impostato su false, l'applicazione ignora l'impostazione globale e il file system viene bloccato durante uno snapshot. Se specificato ma l'applicazione non ha macchine virtuali nella definizione dell'applicazione, l'annotazione viene ignorata. Se non specificato, l'applicazione segue la"impostazione di congelamento globale Trident Protect" .

Se è necessario applicare questa annotazione dopo la creazione di un'applicazione, è possibile utilizzare il seguente comando:

```
kubectl annotate application -n <application CR namespace> <application CR name> protect.trident.netapp.io/skip-vm-freeze="true"
```

+
Esempio YAML:

+

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  annotations:
    protect.trident.netapp.io/skip-vm-freeze: "false"
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: namespace-1
      labelSelector:
        matchLabels:
          app: example-app
    - namespace: namespace-2
      labelSelector:
        matchLabels:
          app: another-example-app
  includedClusterScopedResources:
    - groupVersionKind:
        group: rbac.authorization.k8s.io
        kind: ClusterRole
        version: v1
      labelSelector:
        matchLabels:
          mylabel: test
```

1. (*Facoltativo*) Aggiungi un filtro che includa o escluda le risorse contrassegnate con etichette particolari:

   - **ResourceFilter.resourceSelectionCriteria**: (Necessario per il filtraggio) utilizzare `Include` o includere o `Exclude` escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti parametri resourceMatcher per definire le risorse da includere o escludere:

     - **ResourceFilter.resourceMatchers**: Una matrice di oggetti resourceMatcher. Se si definiscono più elementi in questa matrice, questi corrispondono come un'operazione OR e i campi all'interno di ogni elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.
       - **ResourceMatchers[].group**: (*Optional*) Gruppo della risorsa da filtrare.
       - **ResourceMatchers[].Kind**: (*Optional*) tipo di risorsa da filtrare.
       - **ResourceMatchers[].version**: (*Optional*) versione della risorsa da filtrare.

- **ResourceMatchers[].names**: (*Optional*) nomi nel campo Kubernetes metadata.name della risorsa da filtrare.
- **ResourceMatchers[].namespaces**: (*Optional*) Namespaces nel campo Kubernetes metadata.name della risorsa da filtrare.
- **ResourceMatchers[].labelSelectors**: (*Optional*) stringa del selettore di etichette nel campo Kubernetes metadata.name della risorsa come definito nella "Documentazione Kubernetes" . Ad esempio: `"trident.netapp.io/os=linux"`.

> Quando entrambi `resourceFilter` E `labelSelector` vengono utilizzati, `resourceFilter` corre prima e poi `labelSelector` viene applicato alle risorse risultanti.

Ad esempio:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

2. Dopo aver creato l'applicazione CR per adattarla all'ambiente in uso, applicare il CR. Ad esempio:

```
kubectl apply -f maria-app.yaml
```

**Fasi**

1. Creare e applicare la definizione dell'applicazione utilizzando uno dei seguenti esempi, sostituendo i valori tra parentesi con le informazioni dell'ambiente. È possibile includere spazi dei nomi e risorse nella definizione dell'applicazione utilizzando elenchi separati da virgole con gli argomenti illustrati negli esempi.

   Facoltativamente, quando si crea un'app è possibile utilizzare un'annotazione per specificare se l'applicazione può scrivere sul file system durante uno snapshot. Ciò è applicabile solo alle applicazioni definite da macchine virtuali, come negli ambienti KubeVirt, in cui si verificano blocchi del file system prima degli snapshot. Se si imposta l'annotazione su `true` , l'applicazione ignora l'impostazione globale e può scrivere sul file system durante uno snapshot. Se lo imposti su `false` ,

l'applicazione ignora l'impostazione globale e il file system viene bloccato durante uno snapshot. Se si utilizza l'annotazione ma l'applicazione non ha macchine virtuali nella definizione dell'applicazione, l'annotazione viene ignorata. Se non si utilizza l'annotazione, l'applicazione segue la"impostazione di congelamento globale Trident Protect" .

Per specificare l'annotazione quando si utilizza l'interfaccia CLI per creare un'applicazione, è possibile utilizzare l' `--annotation`indicatore.

- Creare l'applicazione e utilizzare 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>
```

- Creare l'applicazione e configurare l'impostazione dell'applicazione locale per il comportamento di blocco del filesystem:

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-
namespace> --annotation protect.trident.netapp.io/skip-vm-freeze
=<"true"|"false">
```

Puoi usare `--resource-filter-include` E `--resource-filter-exclude` flag per includere o escludere risorse in base a `resourceSelectionCriteria` come gruppo, tipo, versione, etichette, nomi e namespace, come mostrato nel seguente esempio:

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-namespace>
--resource-filter-include
'[{"Group":"apps","Kind":"Deployment","Version":"v1","Names":["my-
deployment"],"Namespaces":["my-
namespace"],"LabelSelectors":["app=my-app"]}]'
```

## Proteggi le applicazioni utilizzando Trident Protect

È possibile proteggere tutte le app gestite da Trident Protect eseguendo snapshot e backup tramite una policy di protezione automatizzata o su base ad hoc.

È possibile configurare Trident Protect in modo che blocchi e sblocchi i file system durante le operazioni di protezione dei dati. "Scopri di più sulla configurazione del congelamento del file system con Trident Protect".

### Crea un'istantanea on-demand

Puoi creare uno snapshot on-demand in qualsiasi momento.

Le risorse soggette a ambito cluster sono incluse in un backup, in uno snapshot o in un clone, se fanno riferimento esplicitamente nella definizione dell'applicazione o se hanno riferimenti a uno qualsiasi dei namespace delle applicazioni.

#### Creare un'istantanea utilizzando una CR

**Fasi**

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-snapshot-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.applicationRef**: Il nome Kubernetes dell'applicazione da snapshot.
   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui devono essere memorizzati i contenuti (metadati) dello snapshot.
   - **Spec.reclaimPolicy**: (*Optional*) definisce cosa accade all'AppArchive di uno snapshot quando lo snapshot CR viene eliminato. Ciò significa che anche se impostato su `Retain`, l'istantanea verrà eliminata. Opzioni valide:
     - `Retain` (impostazione predefinita)
     - `Delete`

```
apiVersion: protect.trident.netapp.io/v1
kind: Snapshot
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  reclaimPolicy: Delete
```

3. Dopo aver popolato il `trident-protect-snapshot-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-snapshot-cr.yaml
```

#### Creare una snapshot utilizzando la CLI

**Fasi**

1. Creare l'istantanea, sostituendo i valori tra parentesi con le informazioni dell'ambiente. Ad esempio:

```
tridentctl-protect create snapshot <my_snapshot_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> -n
<application_namespace>
```

### Crea un backup su richiesta

Puoi eseguire il backup di un'app in qualsiasi momento.

> Le risorse soggette a ambito cluster sono incluse in un backup, in uno snapshot o in un clone, se fanno riferimento esplicitamente nella definizione dell'applicazione o se hanno riferimenti a uno qualsiasi dei namespace delle applicazioni.

#### Prima di iniziare

Assicurati che la scadenza del token di sessione AWS sia sufficiente per eventuali operazioni di backup S3 a esecuzione prolungata. Se il token scade durante l'operazione di backup, l'operazione potrebbe non riuscire.

- Per ulteriori informazioni sulla verifica della scadenza corrente del token di sessione, fare riferimento "Documentazione di API AWS" al .
- Per ulteriori informazioni sulle credenziali con le risorse AWS, fare riferimento al "Documentazione di AWS IAM".

**Creare un backup utilizzando una CR**

**Fasi**

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-backup-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.applicationRef**: (*required*) il nome Kubernetes dell'applicazione di cui eseguire il backup.
   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui devono essere memorizzati i contenuti di backup.
   - **Spec.dataMover**: (*Optional*) stringa che indica quale strumento di backup utilizzare per l'operazione di backup. Valori possibili (distinzione tra maiuscole e minuscole):
     - `Restic`
     - `Kopia` (impostazione predefinita)
   - **Spec.reclaimPolicy**: (*Optional*) definisce cosa accade a un backup quando viene rilasciato dalla relativa dichiarazione. Valori possibili:
     - `Delete`
     - `Retain` (impostazione predefinita)
   - **spec.snapshotRef**: (*Facoltativo*): Nome dello snapshot da utilizzare come origine del backup. Se non viene fornito, verrà creato e eseguito il backup di uno snapshot temporaneo.

     Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: Kopia
```

3. Dopo aver popolato il `trident-protect-backup-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-backup-cr.yaml
```

**Creare un backup utilizzando l'interfaccia CLI**

**Fasi**

1. Creare il backup, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente. Ad esempio:

```
tridentctl-protect create backup <my_backup_name> --appvault <my-vault-name> --app <name_of_app_to_back_up> --data-mover <Kopia_or_Restic> -n <application_namespace>
```

È possibile utilizzare il `--full-backup` flag per specificare se un backup deve essere non incrementale. Per impostazione predefinita, tutti i backup sono incrementali. Quando si utilizza questo indicatore, il backup diventa non incrementale. È consigliabile eseguire periodicamente un backup completo, quindi eseguire backup incrementali tra un backup completo e l'altro, in modo da ridurre al minimo il rischio associato ai ripristini.

#### Annotazioni di backup supportate

Nella tabella seguente vengono descritte le annotazioni che è possibile utilizzare durante la creazione di un CR di backup:

| Annotazione | Tipo | Descrizione | Valore predefinito |
|---|---|---|---|
| protect.trident.netapp.io/full-backup | stringa | Specifica se un backup deve essere non incrementale. Impostato su `true` per creare un backup non incrementale. È consigliabile eseguire periodicamente un backup completo e poi eseguire backup incrementali tra un backup completo e l'altro, per ridurre al minimo i rischi associati ai ripristini. | "falso" |
| protect.trident.netapp.io/snapshot-completion-timeout | stringa | Tempo massimo consentito per il completamento dell'intera operazione di snapshot. | "60 metri" |
| protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout | stringa | Tempo massimo consentito affinché gli snapshot del volume raggiungano lo stato pronto all'uso. | "30 metri" |
| protect.trident.netapp.io/volume-snapshots-created-timeout | stringa | Tempo massimo consentito per la creazione di snapshot del volume. | "5m" |
| protect.trident.netapp.io/pvc-bind-timeout-sec | stringa | Tempo massimo (in secondi) di attesa affinché i nuovi PersistentVolumeClaim (PVC) creati raggiungano il `Bound` fase prima del fallimento delle operazioni. | "1200" (20 minuti) |

### Creare un piano di data Protection

Una policy di protezione protegge un'app creando snapshot, backup o entrambi secondo una pianificazione definita. È possibile scegliere di creare snapshot e backup orari, giornalieri, settimanali e mensili e specificare il numero di copie da conservare. È possibile pianificare un backup completo non incrementale utilizzando l'annotazione full-backup-rule. Per impostazione predefinita, tutti i backup sono incrementali. L'esecuzione periodica di un backup completo, insieme a backup incrementali intermedi, aiuta a ridurre il rischio associato ai

ripristini.

- È possibile creare pianificazioni solo per gli snapshot impostando `backupRetention` a zero e `snapshotRetention` a un valore maggiore di zero. Collocamento `snapshotRetention` a zero significa che tutti i backup pianificati creeranno comunque degli snapshot, ma questi saranno temporanei e verranno eliminati immediatamente dopo il completamento del backup.
- Le risorse soggette a ambito cluster sono incluse in un backup, in uno snapshot o in un clone, se fanno riferimento esplicitamente nella definizione dell'applicazione o se hanno riferimenti a uno qualsiasi dei namespace delle applicazioni.

**Creare una pianificazione utilizzando una CR**

**Fasi**

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-schedule-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.dataMover**: (*Optional*) stringa che indica quale strumento di backup utilizzare per l'operazione di backup. Valori possibili (distinzione tra maiuscole e minuscole):
     - `Restic`
     - `Kopia` (impostazione predefinita)
   - **Spec.applicationRef**: Il nome Kubernetes dell'applicazione di cui eseguire il backup.
   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui devono essere memorizzati i contenuti di backup.
   - **spec.backupRetention**: (*Obbligatorio*) Numero di backup da conservare. Zero indica che non devono essere creati backup (solo snapshot).
   - **spec.backupReclaimPolicy**: (*Facoltativo*) Determina cosa succede a un backup se il CR di backup viene eliminato durante il periodo di conservazione. Dopo il periodo di conservazione, i backup vengono sempre eliminati. Valori possibili (con distinzione tra maiuscole e minuscole):
     - `Retain` (impostazione predefinita)
     - `Delete`
   - **spec.snapshotRetention**: (*Obbligatorio*) Numero di snapshot da conservare. Zero indica che non devono essere creati snapshot.
   - **spec.snapshotReclaimPolicy**: (*Facoltativo*) Determina cosa succede a uno snapshot se il CR dello snapshot viene eliminato durante il periodo di conservazione. Dopo il periodo di conservazione, gli snapshot vengono sempre eliminati. Valori possibili (con distinzione tra maiuscole e minuscole):
     - `Retain`
     - `Delete`(predefinito)
   - **spec.granularity**: frequenza di esecuzione della pianificazione. Valori possibili, insieme ai campi associati obbligatori:
     - `Hourly`(richiede che tu specifichi `spec.minute` )
     - `Daily`(richiede che tu specifichi `spec.minute` E `spec.hour` )
     - `Weekly`(richiede che tu specifichi `spec.minute, spec.hour` , E `spec.dayOfWeek` )
     - `Monthly`(richiede che tu specifichi `spec.minute, spec.hour` , E `spec.dayOfMonth` )
     - `Custom`
   - **spec.dayOfMonth**: (*Facoltativo*) Il giorno del mese (1 - 31) in cui la pianificazione deve essere eseguita. Questo campo è obbligatorio se la granularità è impostata su `Monthly` . Il valore deve essere fornito come stringa.
   - **spec.dayOfWeek**: (*Facoltativo*) Il giorno della settimana (0 - 7) in cui deve essere eseguita la

pianificazione. I valori 0 o 7 indicano domenica. Questo campo è obbligatorio se la granularità è impostata su `Weekly` . Il valore deve essere fornito come stringa.

- **spec.hour**: (*Facoltativo*) L'ora del giorno (0 - 23) in cui la pianificazione deve essere eseguita. Questo campo è obbligatorio se la granularità è impostata su `Daily` , `Weekly` , O `Monthly` . Il valore deve essere fornito come stringa.
- **spec.minute**: (*Facoltativo*) Il minuto dell'ora (0 - 59) in cui la pianificazione deve essere eseguita. Questo campo è obbligatorio se la granularità è impostata su `Hourly` , `Daily` , `Weekly` , O `Monthly` . Il valore deve essere fornito come stringa.

  Esempio di YAML per la pianificazione di backup e snapshot:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: Daily
  hour: "0"
  minute: "0"
```

  Esempio di YAML per la pianificazione solo snapshot:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-snapshot-schedule
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "0"
  snapshotRetention: "15"
  granularity: Daily
  hour: "2"
  minute: "0"
```

3. Dopo aver popolato il `trident-protect-schedule-cr.yaml` file con i valori corretti, applicare la

CR:

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

**Creare una pianificazione utilizzando l'interfaccia CLI**

**Fasi**

1. Creare il programma di protezione, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente. Ad esempio:

> È possibile utilizzare `tridentctl-protect create schedule --help` per visualizzare informazioni dettagliate sulla guida per questo comando.

```
tridentctl-protect create schedule <my_schedule_name> \
  --appvault <my_appvault_name> \
  --app <name_of_app_to_snapshot> \
  --backup-retention <how_many_backups_to_retain> \
  --backup-reclaim-policy <Retain|Delete (default Retain)> \
  --data-mover <Kopia_or_Restic> \
  --day-of-month <day_of_month_to_run_schedule> \
  --day-of-week <day_of_week_to_run_schedule> \
  --granularity <frequency_to_run> \
  --hour <hour_of_day_to_run> \
  --minute <minute_of_hour_to_run> \
  --recurrence-rule <recurrence> \
  --snapshot-retention <how_many_snapshots_to_retain> \
  --snapshot-reclaim-policy <Retain|Delete (default Delete)> \
  --full-backup-rule <string> \
  --run-immediately <true|false> \
  -n <application_namespace>
```

I seguenti flag forniscono un controllo aggiuntivo sulla tua pianificazione:

- **Pianificazione del backup completo**: utilizzare `--full-backup-rule` flag per pianificare backup completi non incrementali. Questa bandiera funziona solo con `--granularity Daily`. Valori possibili:
  - `Always`: Crea un backup completo ogni giorno.
  - Giorni feriali specifici: specificare uno o più giorni separati da virgole (ad esempio, `"Monday,Thursday"`). Valori validi: lunedì, martedì, mercoledì, giovedì, venerdì, sabato, domenica.

> IL `--full-backup-rule` il flag non funziona con la granularità oraria, settimanale o mensile.

- **Programmazioni solo snapshot**: Imposta `--backup-retention 0` e specificare un valore maggiore di zero per `--snapshot-retention`.

#### Annotazioni di pianificazione supportate

Nella tabella seguente vengono descritte le annotazioni che è possibile utilizzare durante la creazione di una CR di pianificazione:

| Annotazione | Tipo | Descrizione | Valore predefinito |
|---|---|---|---|
| protect.trident.netapp.io/full-backup-rule | stringa | Specifica la regola per la pianificazione dei backup completi. Puoi impostarlo su `Always` per un backup completo costante o personalizzarlo in base alle tue esigenze. Ad esempio, se si sceglie la granularità giornaliera, è possibile specificare i giorni feriali in cui deve essere eseguito il backup completo (ad esempio, `"Monday,Thursday"`). I valori validi per i giorni feriali sono: lunedì, martedì, mercoledì, giovedì, venerdì, sabato, domenica. Si noti che questa annotazione può essere utilizzata solo con pianificazioni che hanno `granularity` impostato su `Daily`. | Non impostato (tutti i backup sono incrementali) |
| protect.trident.netapp.io/snapshot-completion-timeout | stringa | Tempo massimo consentito per il completamento dell'intera operazione di snapshot. | "60 metri" |
| protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout | stringa | Tempo massimo consentito affinché gli snapshot del volume raggiungano lo stato pronto all'uso. | "30 metri" |
| protect.trident.netapp.io/volume-snapshots-created-timeout | stringa | Tempo massimo consentito per la creazione di snapshot del volume. | "5m" |
| protect.trident.netapp.io/pvc-bind-timeout-sec | stringa | Tempo massimo (in secondi) di attesa affinché i nuovi PersistentVolumeClaim (PVC) creati raggiungano il `Bound` fase prima del fallimento delle operazioni. | "1200" (20 minuti) |

### Eliminare uno snapshot

Eliminare le snapshot pianificate o on-demand non più necessarie.

#### Fasi

1. Rimuovere l'istantanea CR associata all'istantanea:

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

### Eliminare un backup

Eliminare i backup pianificati o on-demand non più necessari.

Assicurati che la politica di recupero sia impostata su `Delete` per rimuovere tutti i dati di backup dall'archiviazione degli oggetti. L'impostazione predefinita del criterio è `Retain` per evitare la perdita accidentale di dati. Se la politica non viene modificata in `Delete` , i dati di backup rimarranno nell'archivio oggetti e richiederanno l'eliminazione manuale.

**Fasi**

1. Rimuovere il CR di backup associato al backup:

```
kubectl delete backup <backup_name> -n my-app-namespace
```

### Controllare lo stato di un'operazione di backup

È possibile utilizzare la riga di comando per verificare lo stato di un'operazione di backup in corso, completata o non riuscita.

**Fasi**

1. Utilizzare il seguente comando per recuperare lo stato dell'operazione di backup, sostituendo i valori nei brackes con le informazioni dal proprio ambiente:

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath
='{.status}'
```

### Abilitare backup e ripristino per operazioni Azure-NetApp-Files (ANF)

Se hai installato Trident Protect, puoi abilitare la funzionalità di backup e ripristino a basso consumo di spazio per i backend di archiviazione che utilizzano la classe di archiviazione azure-netapp-files e sono stati creati prima di Trident 24.06. Questa funzionalità funziona con volumi NFSv4 e non consuma spazio aggiuntivo dal pool di capacità.

**Prima di iniziare**

Verificare quanto segue:

- Hai installato Trident Protect.
- Hai definito un'applicazione in Trident Protect. Questa applicazione avrà funzionalità di protezione limitate finché non avrai completato questa procedura.
- È stata `azure-netapp-files` selezionata come classe di archiviazione predefinita per il backend di archiviazione.

**Espandere per la procedura di configurazione**

1. Se il volume ANF è stato creato prima dell'aggiornamento a Trident 24,10, procedere come segue in Trident:

   a. Abilitare la directory snapshot per ogni PV basata su file Azure-NetApp e associata all'applicazione:

   ```
   tridentctl update volume <pv name> --snapshot-dir=true -n trident
   ```

   b. Confermare che la directory snapshot è stata abilitata per ogni PV associato:

   ```
   tridentctl get volume <pv name> -n trident -o yaml | grep
   snapshotDir
   ```

   Risposta:

   ```
   snapshotDirectory: "true"
   ```

   +
   Quando la directory degli snapshot non è abilitata, Trident Protect sceglie la funzionalità di backup normale, che consuma temporaneamente spazio nel pool di capacità durante il processo di backup. In questo caso, assicurarsi che nel pool di capacità sia disponibile spazio sufficiente per creare un volume temporaneo delle dimensioni del volume sottoposto a backup.

**Risultato**

L'applicazione è pronta per il backup e il ripristino tramite Trident Protect. Ogni PVC può essere utilizzato anche da altre applicazioni per backup e ripristini.

## Ripristino delle applicazioni

### Ripristina le applicazioni utilizzando Trident Protect

Puoi utilizzare Trident Protect per ripristinare la tua applicazione da uno snapshot o da un backup. Il ripristino da uno snapshot esistente sarà più rapido se si ripristina l'applicazione nello stesso cluster.

- Quando si ripristina un'applicazione, tutti i collegamenti di esecuzione configurati per l'applicazione vengono ripristinati con l'applicazione. Se è presente un gancio di esecuzione post-ripristino, viene eseguito automaticamente come parte dell'operazione di ripristino.
- Il ripristino da un backup a un namespace diverso o al namespace originale è supportato per i volumi qtree. Tuttavia, il ripristino da uno snapshot a un namespace diverso o al namespace originale non è supportato per i volumi qtree.
- È possibile utilizzare le impostazioni avanzate per personalizzare le operazioni di ripristino. Per saperne di più, fare riferimento a "Utilizza le impostazioni di ripristino avanzate Trident Protect".

#### Ripristino da un backup a uno spazio dei nomi diverso

Quando si ripristina un backup in uno spazio dei nomi diverso utilizzando un CR BackupRestore, Trident Protect ripristina l'applicazione in un nuovo spazio dei nomi e crea un CR dell'applicazione per l'applicazione ripristinata. Per proteggere l'applicazione ripristinata, creare backup o snapshot su richiesta oppure stabilire una pianificazione di protezione.

- Il ripristino di un backup in uno spazio dei nomi diverso con le risorse esistenti non altererà le risorse che condividono i nomi con quelli del backup. Per ripristinare tutte le risorse del backup, eliminare e ricreare lo spazio dei nomi di destinazione o ripristinare il backup in un nuovo spazio dei nomi.
- Quando si utilizza una CR per ripristinare un nuovo namespace, è necessario creare manualmente il namespace di destinazione prima di applicare la CR. Trident Protect crea automaticamente gli spazi dei nomi solo quando si utilizza la CLI.

#### Prima di iniziare

Assicurati che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino S3 con esecuzione prolungata. Se il token scade durante l'operazione di ripristino, l'operazione potrebbe non riuscire.

- Per ulteriori informazioni sulla verifica della scadenza corrente del token di sessione, fare riferimento "Documentazione di API AWS" al .
- Per ulteriori informazioni sulle credenziali con le risorse AWS, fare riferimento al "Documentazione di AWS IAM".

Quando si ripristinano i backup utilizzando Kopia come strumento di spostamento dati, è possibile specificare facoltativamente annotazioni nel CR o tramite la CLI per controllare il comportamento dell'archiviazione temporanea utilizzata da Kopia. Fare riferimento al "Documentazione Kopia" per maggiori informazioni sulle opzioni che puoi configurare. Utilizzare il `tridentctl-protect create --help` comando per ulteriori informazioni sulla specifica delle annotazioni con la CLI Trident Protect.

**Utilizzare un CR**

**Fasi**

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-backup-restore-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono memorizzati i contenuti di backup. Per trovare il percorso, utilizzare il seguente comando:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui sono memorizzati i contenuti di backup.
   - **spec.namespaceMapping**: mappatura dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituire `my-source-namespace` e `my-destination-namespace` con le informazioni del proprio ambiente.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: BackupRestore
     metadata:
       name: my-cr-name
       namespace: my-destination-namespace
     spec:
       appArchivePath: my-backup-path
       appVaultRef: appvault-name
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. (*Optional*) se è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda risorse contrassegnate con determinate etichette:

   > Trident Protect seleziona automaticamente alcune risorse in base alla loro relazione con le risorse selezionate. Ad esempio, se selezioni una risorsa di richiesta di volume persistente e a questa è associato un pod, Trident Protect ripristinerà anche il pod associato.

   - **ResourceFilter.resourceSelectionCriteria**: (Necessario per il filtraggio) utilizzare `Include` o includere o `Exclude` escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti parametri resourceMatcher per definire le risorse da includere o escludere:
     - **ResourceFilter.resourceMatchers**: Una matrice di oggetti resourceMatcher. Se si definiscono più elementi in questa matrice, questi corrispondono come un'operazione OR e i campi all'interno di ogni elemento (gruppo, tipo, versione) corrispondono come un'operazione

AND.

- **ResourceMatchers[].group**: (*Optional*) Gruppo della risorsa da filtrare.
- **ResourceMatchers[].Kind**: (*Optional*) tipo di risorsa da filtrare.
- **ResourceMatchers[].version**: (*Optional*) versione della risorsa da filtrare.
- **ResourceMatchers[].names**: (*Optional*) nomi nel campo Kubernetes metadata.name della risorsa da filtrare.
- **ResourceMatchers[].namespaces**: (*Optional*) Namespaces nel campo Kubernetes metadata.name della risorsa da filtrare.
- **ResourceMatchers[].labelSelectors**: (*Optional*) stringa del selettore di etichette nel campo Kubernetes metadata.name della risorsa come definito nella "Documentazione Kubernetes" . Ad esempio: `"trident.netapp.io/os=linux"`.

  Ad esempio:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Dopo aver popolato il `trident-protect-backup-restore-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

**Utilizzare la CLI**

**Fasi**

1. Ripristinare il backup su uno spazio dei nomi diverso, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente. L' `namespace-mapping`argomento utilizza spazi dei nomi separati da due punti per mappare gli spazi dei nomi di origine agli spazi dei nomi di destinazione corretti nel formato `source1:dest1,source2:dest2`. Ad esempio:

```
tridentctl-protect create backuprestore <my_restore_name> \
--backup <backup_namespace>/<backup_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
-n <application_namespace>
```

#### Eseguire il ripristino da un backup nello spazio dei nomi originale

È possibile ripristinare un backup nello spazio dei nomi originale in qualsiasi momento.

**Prima di iniziare**

Assicurati che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino S3 con esecuzione prolungata. Se il token scade durante l'operazione di ripristino, l'operazione potrebbe non riuscire.

- Per ulteriori informazioni sulla verifica della scadenza corrente del token di sessione, fare riferimento "Documentazione di API AWS" al .
- Per ulteriori informazioni sulle credenziali con le risorse AWS, fare riferimento al "Documentazione di AWS IAM".

> Quando si ripristinano i backup utilizzando Kopia come strumento di spostamento dati, è possibile specificare facoltativamente annotazioni nel CR o tramite la CLI per controllare il comportamento dell'archiviazione temporanea utilizzata da Kopia. Fare riferimento al "Documentazione Kopia" per maggiori informazioni sulle opzioni che puoi configurare. Utilizzare il `tridentctl-protect create --help` comando per ulteriori informazioni sulla specifica delle annotazioni con la CLI Trident Protect.

**Utilizzare un CR**

**Fasi**

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-backup-ipr-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono memorizzati i contenuti di backup. Per trovare il percorso, utilizzare il seguente comando:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui sono memorizzati i contenuti di backup.

     Ad esempio:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appArchivePath: my-backup-path
  appVaultRef: appvault-name
```

3. (*Optional*) se è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda risorse contrassegnate con determinate etichette:

   Trident Protect seleziona automaticamente alcune risorse in base alla loro relazione con le risorse selezionate. Ad esempio, se selezioni una risorsa di richiesta di volume persistente e a questa è associato un pod, Trident Protect ripristinerà anche il pod associato.

   - **ResourceFilter.resourceSelectionCriteria**: (Necessario per il filtraggio) utilizzare `Include` o includere o `Exclude` escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti parametri resourceMatcher per definire le risorse da includere o escludere:
     - **ResourceFilter.resourceMatchers**: Una matrice di oggetti resourceMatcher. Se si definiscono più elementi in questa matrice, questi corrispondono come un'operazione OR e i campi all'interno di ogni elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.
       - **ResourceMatchers[].group**: (*Optional*) Gruppo della risorsa da filtrare.
       - **ResourceMatchers[].Kind**: (*Optional*) tipo di risorsa da filtrare.

- **ResourceMatchers[].version**: (*Optional*) versione della risorsa da filtrare.
- **ResourceMatchers[].names**: (*Optional*) nomi nel campo Kubernetes metadata.name della risorsa da filtrare.
- **ResourceMatchers[].namespaces**: (*Optional*) Namespaces nel campo Kubernetes metadata.name della risorsa da filtrare.
- **ResourceMatchers[].labelSelectors**: (*Optional*) stringa del selettore di etichette nel campo Kubernetes metadata.name della risorsa come definito nella "Documentazione Kubernetes" . Ad esempio: `"trident.netapp.io/os=linux"`.

  Ad esempio:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Dopo aver popolato il `trident-protect-backup-ipr-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

**Utilizzare la CLI**

**Fasi**

1. Ripristinare il backup nello spazio dei nomi originale, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente. L' `backup`argomento utilizza uno spazio dei nomi e un nome di backup nel formato `<namespace>/<name>`. Ad esempio:

```
tridentctl-protect create backupinplacerestore <my_restore_name> \
--backup <namespace/backup_to_restore> \
-n <application_namespace>
```

#### Ripristino da un backup a un cluster diverso

In caso di problemi con il cluster originale, è possibile ripristinare un backup su un cluster diverso.

- Quando si ripristinano i backup utilizzando Kopia come strumento di spostamento dati, è possibile specificare facoltativamente annotazioni nel CR o tramite la CLI per controllare il comportamento dell'archiviazione temporanea utilizzata da Kopia. Fare riferimento al "Documentazione Kopia" per maggiori informazioni sulle opzioni che puoi configurare. Utilizzare il `tridentctl-protect create --help` comando per ulteriori informazioni sulla specifica delle annotazioni con la CLI Trident Protect.
- Quando si utilizza una CR per ripristinare un nuovo namespace, è necessario creare manualmente il namespace di destinazione prima di applicare la CR. Trident Protect crea automaticamente gli spazi dei nomi solo quando si utilizza la CLI.

**Prima di iniziare**

Assicurarsi che siano soddisfatti i seguenti prerequisiti:

- Nel cluster di destinazione è installato Trident Protect.
- Il cluster di destinazione ha accesso al percorso bucket dello stesso AppVault del cluster di origine, dove è memorizzato il backup.
- Assicurarsi che l'ambiente locale possa connettersi al bucket di archiviazione degli oggetti definito in AppVault CR durante l'esecuzione di `tridentctl-protect get appvaultcontent` comando. Se le restrizioni di rete impediscono l'accesso, eseguire invece la CLI Trident Protect da un pod sul cluster di destinazione.
- Assicurati che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino con esecuzione prolungata. Se il token scade durante l'operazione di ripristino, l'operazione potrebbe non riuscire.
  - Per ulteriori informazioni sulla verifica della scadenza corrente del token di sessione, fare riferimento "Documentazione di API AWS" al .
  - Per ulteriori informazioni sulle credenziali con le risorse AWS, fare riferimento al "Documentazione AWS".

**Fasi**

1. Verificare la disponibilità di AppVault CR sul cluster di destinazione utilizzando il plug-in Trident Protect CLI:

```
tridentctl-protect get appvault --context <destination_cluster_name>
```

   Verificare che lo spazio dei nomi destinato al ripristino dell'applicazione esista nel cluster di destinazione.

2. Visualizzare il contenuto di backup dell'AppVault disponibile dal cluster di destinazione:

```
tridentctl-protect get appvaultcontent <appvault_name> \
--show-resources backup \
--show-paths \
--context <destination_cluster_name>
```

L’esecuzione di questo comando visualizza i backup disponibili in AppVault, inclusi i relativi cluster di origine, i nomi delle applicazioni corrispondenti, i timestamp e i percorsi di archivio.

**Esempio di output:**

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   CLUSTER   |    APP    |  TYPE  |      NAME       |        TIMESTAMP
|    PATH     |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
| production1 | wordpress | backup | wordpress-bkup-1| 2024-10-30
08:37:40 (UTC)| backuppath1 |
| production1 | wordpress | backup | wordpress-bkup-2| 2024-10-30
08:37:40 (UTC)| backuppath2 |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
```

3. Ripristinare l’applicazione nel cluster di destinazione utilizzando il nome AppVault e il percorso di archiviazione:

**Utilizzare un CR**

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-backup-restore-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui sono memorizzati i contenuti di backup.
   - **Spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono memorizzati i contenuti di backup. Per trovare il percorso, utilizzare il seguente comando:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

     Se BackupRestore CR non è disponibile, è possibile utilizzare il comando menzionato al passaggio 2 per visualizzare il contenuto del backup.

   - **spec.namespaceMapping**: mappatura dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituire `my-source-namespace` e `my-destination-namespace` con le informazioni del proprio ambiente.

     Ad esempio:

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: my-cr-name
  namespace: my-destination-namespace
spec:
  appVaultRef: appvault-name
  appArchivePath: my-backup-path
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. Dopo aver popolato il `trident-protect-backup-restore-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

**Utilizzare la CLI**

1. Utilizzare il seguente comando per ripristinare l'applicazione, sostituendo i valori tra parentesi con le informazioni dell'ambiente. L'argomento namespace-mapping utilizza spazi dei nomi separati da due punti per mappare gli spazi dei nomi di origine agli spazi dei nomi di destinazione corretti nel formato

source1:dest1,source2:dest2. Ad esempio:

```
tridentctl-protect create backuprestore <restore_name> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--appvault <appvault_name> \
--path <backup_path> \
--context <destination_cluster_name> \
-n <application_namespace>
```

#### Ripristino da uno snapshot a uno spazio dei nomi diverso

È possibile ripristinare i dati da uno snapshot utilizzando un file di risorse personalizzato (CR) in uno spazio dei nomi diverso o nello spazio dei nomi di origine originale. Quando si ripristina uno snapshot in un namespace diverso utilizzando un CR SnapshotRestore, Trident Protect ripristina l'applicazione in un nuovo namespace e crea un CR per l'applicazione ripristinata. Per proteggere l'applicazione ripristinata, creare backup o snapshot su richiesta oppure stabilire una pianificazione di protezione.

- SnapshotRestore supporta il `spec.storageClassMapping` attributo, ma solo quando le classi di archiviazione di origine e di destinazione utilizzano lo stesso backend di archiviazione. Se si tenta di ripristinare un `StorageClass` che utilizza un backend di archiviazione diverso, l'operazione di ripristino non riuscirà.
- Quando si utilizza una CR per ripristinare un nuovo namespace, è necessario creare manualmente il namespace di destinazione prima di applicare la CR. Trident Protect crea automaticamente gli spazi dei nomi solo quando si utilizza la CLI.

**Prima di iniziare**

Assicurati che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino S3 con esecuzione prolungata. Se il token scade durante l'operazione di ripristino, l'operazione potrebbe non riuscire.

- Per ulteriori informazioni sulla verifica della scadenza corrente del token di sessione, fare riferimento "Documentazione di API AWS" al .
- Per ulteriori informazioni sulle credenziali con le risorse AWS, fare riferimento al "Documentazione di AWS IAM".

**Utilizzare un CR**

**Fasi**

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-snapshot-restore-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui sono memorizzati i contenuti dello snapshot.
   - **Spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono memorizzati i contenuti dello snapshot. Per trovare il percorso, utilizzare il seguente comando:

     ```
     kubectl get snapshots <SNAPHOT_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.namespaceMapping**: mappatura dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituire `my-source-namespace` e `my-destination-namespace` con le informazioni del proprio ambiente.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. (*Optional*) se è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda risorse contrassegnate con determinate etichette:

   > Trident Protect seleziona automaticamente alcune risorse in base alla loro relazione con le risorse selezionate. Ad esempio, se selezioni una risorsa di richiesta di volume persistente e a questa è associato un pod, Trident Protect ripristinerà anche il pod associato.

   - **ResourceFilter.resourceSelectionCriteria**: (Necessario per il filtraggio) utilizzare `Include` o includere o `Exclude` escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti parametri resourceMatcher per definire le risorse da includere o escludere:
     - **ResourceFilter.resourceMatchers**: Una matrice di oggetti resourceMatcher. Se si definiscono più elementi in questa matrice, questi corrispondono come un'operazione OR e i

campi all'interno di ogni elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.

- **ResourceMatchers[].group**: (*Optional*) Gruppo della risorsa da filtrare.
- **ResourceMatchers[].Kind**: (*Optional*) tipo di risorsa da filtrare.
- **ResourceMatchers[].version**: (*Optional*) versione della risorsa da filtrare.
- **ResourceMatchers[].names**: (*Optional*) nomi nel campo Kubernetes metadata.name della risorsa da filtrare.
- **ResourceMatchers[].namespaces**: (*Optional*) Namespaces nel campo Kubernetes metadata.name della risorsa da filtrare.
- **ResourceMatchers[].labelSelectors**: (*Optional*) stringa del selettore di etichette nel campo Kubernetes metadata.name della risorsa come definito nella "Documentazione Kubernetes" . Ad esempio: `"trident.netapp.io/os=linux"`.

  Ad esempio:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Dopo aver popolato il `trident-protect-snapshot-restore-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

**Utilizzare la CLI**

**Fasi**

1. Ripristinare lo snapshot in uno spazio dei nomi diverso, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente.

   - L'`snapshot`argomento utilizza uno spazio dei nomi e un nome snapshot nel formato `<namespace>/<name>.

- L'`namespace-mapping`argomento utilizza spazi dei nomi separati da due punti per mappare gli spazi dei nomi di origine agli spazi dei nomi di destinazione corretti nel formato `source1:dest1,source2:dest2.

  Ad esempio:

```
tridentctl-protect create snapshotrestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
-n <application_namespace>
```

#### Ripristinare da uno snapshot allo spazio dei nomi originale

È possibile ripristinare uno snapshot nello spazio dei nomi originale in qualsiasi momento.

**Prima di iniziare**

Assicurati che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino S3 con esecuzione prolungata. Se il token scade durante l'operazione di ripristino, l'operazione potrebbe non riuscire.

- Per ulteriori informazioni sulla verifica della scadenza corrente del token di sessione, fare riferimento "Documentazione di API AWS" al .
- Per ulteriori informazioni sulle credenziali con le risorse AWS, fare riferimento al "Documentazione di AWS IAM".

**Utilizzare un CR**

**Fasi**

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-snapshot-ipr-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui sono memorizzati i contenuti dello snapshot.
   - **Spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono memorizzati i contenuti dello snapshot. Per trovare il percorso, utilizzare il seguente comando:

     ```
     kubectl get snapshots <SNAPSHOT_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotInplaceRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
     ```

3. (*Optional*) se è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda risorse contrassegnate con determinate etichette:

   > Trident Protect seleziona automaticamente alcune risorse in base alla loro relazione con le risorse selezionate. Ad esempio, se selezioni una risorsa di richiesta di volume persistente e a questa è associato un pod, Trident Protect ripristinerà anche il pod associato.

   - **ResourceFilter.resourceSelectionCriteria**: (Necessario per il filtraggio) utilizzare `Include` o includere o `Exclude` escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti parametri resourceMatcher per definire le risorse da includere o escludere:
     - **ResourceFilter.resourceMatchers**: Una matrice di oggetti resourceMatcher. Se si definiscono più elementi in questa matrice, questi corrispondono come un'operazione OR e i campi all'interno di ogni elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.
       - **ResourceMatchers[].group**: (*Optional*) Gruppo della risorsa da filtrare.
       - **ResourceMatchers[].Kind**: (*Optional*) tipo di risorsa da filtrare.
       - **ResourceMatchers[].version**: (*Optional*) versione della risorsa da filtrare.

- **ResourceMatchers[].names**: (*Optional*) nomi nel campo Kubernetes metadata.name della risorsa da filtrare.
- **ResourceMatchers[].namespaces**: (*Optional*) Namespaces nel campo Kubernetes metadata.name della risorsa da filtrare.
- **ResourceMatchers[].labelSelectors**: (*Optional*) stringa del selettore di etichette nel campo Kubernetes metadata.name della risorsa come definito nella "Documentazione Kubernetes" . Ad esempio: `"trident.netapp.io/os=linux"`.

  Ad esempio:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Dopo aver popolato il `trident-protect-snapshot-ipr-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

**Utilizzare la CLI**

**Fasi**

1. Ripristinare lo snapshot nello spazio dei nomi originale, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente. Ad esempio:

```
tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
-n <application_namespace>
```

#### Controllare lo stato di un'operazione di ripristino

È possibile utilizzare la riga di comando per verificare lo stato di un'operazione di ripristino in corso, completata o non riuscita.

**Fasi**

1. Utilizzare il seguente comando per recuperare lo stato dell'operazione di ripristino, sostituendo i valori nei brackes con le informazioni dall'ambiente in uso:

```
kubectl get backuprestore -n <namespace_name> <my_restore_cr_name> -o
jsonpath='{.status}'
```

### Utilizza le impostazioni di ripristino avanzate Trident Protect

È possibile personalizzare le operazioni di ripristino utilizzando impostazioni avanzate quali annotazioni, impostazioni dello spazio dei nomi e opzioni di archiviazione per soddisfare esigenze specifiche.

#### Annotazioni ed etichette del namespace durante le operazioni di ripristino e failover

Durante le operazioni di ripristino e failover, vengono applicate etichette e annotazioni nel namespace di destinazione in modo che corrispondano alle etichette e alle annotazioni nel namespace di origine. Vengono aggiunte etichette o annotazioni dallo spazio dei nomi di origine che non esistono nello spazio dei nomi di destinazione e le etichette o annotazioni già esistenti vengono sovrascritte per corrispondere al valore dello spazio dei nomi di origine. Le etichette o le annotazioni presenti solo nello spazio dei nomi di destinazione rimangono invariate.

> Se si utilizza Red Hat OpenShift, è importante tenere presente il ruolo fondamentale delle annotazioni dello spazio dei nomi negli ambienti OpenShift. Le annotazioni dello spazio dei nomi garantiscono che i pod ripristinati aderiscano alle autorizzazioni appropriate e alle configurazioni di sicurezza definite dai vincoli del contesto di sicurezza (SCC) di OpenShift e possano accedere ai volumi senza problemi di autorizzazione. Per maggiori informazioni, fare riferimento al"Documentazione dei vincoli del contesto di protezione OpenShift" .

Puoi impedire la sovrascrittura delle annotazioni specifiche nel namespace di destinazione impostando la variabile dell'ambiente Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` prima di eseguire l'operazione di ripristino o failover. Ad esempio:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

Quando si esegue un'operazione di ripristino o failover, tutte le annotazioni e le etichette dello spazio dei nomi specificate in `restoreSkipNamespaceAnnotations` E `restoreSkipNamespaceLabels` sono esclusi dall'operazione di ripristino o failover. Assicurarsi che queste impostazioni siano configurate durante l'installazione iniziale di Helm. Per saperne di più, fare riferimento a "Configurare le impostazioni aggiuntive del grafico del timone Trident Protect".

Se hai installato l'applicazione sorgente utilizzando Helm con `--create-namespace` bandiera, un trattamento speciale è riservato al `name` etichetta chiave. Durante il processo di ripristino o failover, Trident Protect copia questa etichetta nello spazio dei nomi di destinazione, ma aggiorna il valore al valore dello spazio dei nomi di destinazione se il valore dell'origine corrisponde allo spazio dei nomi di origine. Se questo valore non corrisponde allo spazio dei nomi di origine, viene copiato nello spazio dei nomi di destinazione senza modifiche.

### Esempio

Nell'esempio seguente viene presentato uno spazio dei nomi di origine e destinazione, ciascuno con annotazioni ed etichette diverse. È possibile visualizzare lo stato dello spazio dei nomi di destinazione prima e dopo l'operazione e il modo in cui le annotazioni e le etichette vengono combinate o sovrascritte nello spazio dei nomi di destinazione.

### Prima dell'operazione di ripristino o failover

La tabella seguente illustra lo stato degli spazi dei nomi di origine e di destinazione di esempio prima dell'operazione di ripristino o failover:

| Namespace | Annotazioni | Etichette |
|---|---|---|
| Namespace ns-1 (origine) | • annotation.one/key: "updatedvalue"<br>• annotation.two/key: "true" | • ambiente=produzione<br>• conformità=hipaa<br>• name=ns-1 |
| Namespace ns-2 (destinazione) | • annotation.one/key: "true"<br>• annotation.three/key: "false" | • ruolo=database |

### Dopo l'operazione di ripristino

La tabella seguente illustra lo stato dello spazio dei nomi di destinazione di esempio dopo l'operazione di ripristino o failover. Alcune chiavi sono state aggiunte, altre sono state sovrascritte e l' `name`etichetta è stata aggiornata per corrispondere allo spazio dei nomi di destinazione:

| Namespace | Annotazioni | Etichette |
|---|---|---|
| Namespace ns-2 (destinazione) | • annotation.one/key: "updatedvalue"<br>• annotation.two/key: "true"<br>• annotation.three/key: "false" | • name=ns-2<br>• conformità=hipaa<br>• ambiente=produzione<br>• ruolo=database |

#### Campi supportati

Questa sezione descrive i campi aggiuntivi disponibili per le operazioni di ripristino.

#### Mappatura delle classi di archiviazione

IL `spec.storageClassMapping` L'attributo definisce una mappatura da una classe di archiviazione presente nell'applicazione di origine a una nuova classe di archiviazione nel cluster di destinazione. È possibile utilizzarlo durante la migrazione di applicazioni tra cluster con classi di archiviazione diverse o quando si modifica il backend di archiviazione per le operazioni BackupRestore.

**Esempio:**

```
storageClassMapping:
  - destination: "destinationStorageClass1"
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

#### Annotazioni supportate

Questa sezione elenca le annotazioni supportate per la configurazione di vari comportamenti nel sistema. Se un'annotazione non viene impostata esplicitamente dall'utente, il sistema utilizzerà il valore predefinito.

| Annotazione | Tipo | Descrizione | Valore predefinito |
|---|---|---|---|
| proteggi.trident.netapp.io/data-mover-timeout-sec | stringa | Tempo massimo (in secondi) consentito per l'interruzione dell'operazione di spostamento dei dati. | "300" |
| protect.trident.netapp.io/kopia-content-cache-size-limit-mb | stringa | Limite massimo di dimensione (in megabyte) per la cache dei contenuti di Kopia. | "1000" |
| protect.trident.netapp.io/pvc-bind-timeout-sec | stringa | Tempo massimo (in secondi) di attesa affinché i nuovi PersistentVolumeClaim (PVC) creati raggiungano il `Bound` fase prima del fallimento delle operazioni. Si applica a tutti i tipi di ripristino CR (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Utilizzare un valore più alto se il backend di archiviazione o il cluster richiedono spesso più tempo. | "1200" (20 minuti) |

## Replicare le applicazioni utilizzando NetApp SnapMirror e Trident Protect

Utilizzando Trident Protect, è possibile sfruttare le funzionalità di replica asincrona della tecnologia NetApp SnapMirror per replicare le modifiche dei dati e delle applicazioni da un backend di storage all'altro, sullo stesso cluster o tra cluster diversi.

### Annotazioni ed etichette del namespace durante le operazioni di ripristino e failover

Durante le operazioni di ripristino e failover, vengono applicate etichette e annotazioni nel namespace di destinazione in modo che corrispondano alle etichette e alle annotazioni nel namespace di origine. Vengono aggiunte etichette o annotazioni dallo spazio dei nomi di origine che non esistono nello spazio dei nomi di destinazione e le etichette o annotazioni già esistenti vengono sovrascritte per corrispondere al valore dello spazio dei nomi di origine. Le etichette o le annotazioni presenti solo nello spazio dei nomi di destinazione rimangono invariate.

> Se si utilizza Red Hat OpenShift, è importante tenere presente il ruolo fondamentale delle annotazioni dello spazio dei nomi negli ambienti OpenShift. Le annotazioni dello spazio dei nomi garantiscono che i pod ripristinati aderiscano alle autorizzazioni appropriate e alle configurazioni di sicurezza definite dai vincoli del contesto di sicurezza (SCC) di OpenShift e possano accedere ai volumi senza problemi di autorizzazione. Per maggiori informazioni, fare riferimento al"Documentazione dei vincoli del contesto di protezione OpenShift" .

Puoi impedire la sovrascrittura delle annotazioni specifiche nel namespace di destinazione impostando la variabile dell'ambiente Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` prima di eseguire l'operazione di ripristino o failover. Ad esempio:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

> Quando si esegue un'operazione di ripristino o failover, tutte le annotazioni e le etichette dello spazio dei nomi specificate in `restoreSkipNamespaceAnnotations` E `restoreSkipNamespaceLabels` sono esclusi dall'operazione di ripristino o failover. Assicurarsi che queste impostazioni siano configurate durante l'installazione iniziale di Helm. Per saperne di più, fare riferimento a "Configurare le impostazioni aggiuntive del grafico del timone Trident Protect".

Se hai installato l'applicazione sorgente utilizzando Helm con `--create-namespace` bandiera, un trattamento speciale è riservato al `name` etichetta chiave. Durante il processo di ripristino o failover, Trident Protect copia questa etichetta nello spazio dei nomi di destinazione, ma aggiorna il valore al valore dello spazio dei nomi di destinazione se il valore dell'origine corrisponde allo spazio dei nomi di origine. Se questo valore non corrisponde allo spazio dei nomi di origine, viene copiato nello spazio dei nomi di destinazione senza modifiche.

#### Esempio

Nell'esempio seguente viene presentato uno spazio dei nomi di origine e destinazione, ciascuno con annotazioni ed etichette diverse. È possibile visualizzare lo stato dello spazio dei nomi di destinazione prima e dopo l'operazione e il modo in cui le annotazioni e le etichette vengono combinate o sovrascritte nello spazio dei nomi di destinazione.

### Prima dell'operazione di ripristino o failover

La tabella seguente illustra lo stato degli spazi dei nomi di origine e di destinazione di esempio prima dell'operazione di ripristino o failover:

| Namespace | Annotazioni | Etichette |
|---|---|---|
| Namespace ns-1 (origine) | • annotation.one/key: "updatedvalue"<br>• annotation.two/key: "true" | • ambiente=produzione<br>• conformità=hipaa<br>• name=ns-1 |
| Namespace ns-2 (destinazione) | • annotation.one/key: "true"<br>• annotation.three/key: "false" | • ruolo=database |

### Dopo l'operazione di ripristino

La tabella seguente illustra lo stato dello spazio dei nomi di destinazione di esempio dopo l'operazione di ripristino o failover. Alcune chiavi sono state aggiunte, altre sono state sovrascritte e l' `name`etichetta è stata aggiornata per corrispondere allo spazio dei nomi di destinazione:

| Namespace | Annotazioni | Etichette |
|---|---|---|
| Namespace ns-2 (destinazione) | • annotation.one/key: "updatedvalue"<br>• annotation.two/key: "true"<br>• annotation.three/key: "false" | • name=ns-2<br>• conformità=hipaa<br>• ambiente=produzione<br>• ruolo=database |

È possibile configurare Trident Protect in modo che blocchi e sblocchi i file system durante le operazioni di protezione dei dati. "Scopri di più sulla configurazione del congelamento del file system con Trident Protect".

### Hook di esecuzione durante le operazioni di failover e reverse

Quando si utilizza la relazione AppMirror per proteggere l'applicazione, ci sono comportamenti specifici relativi agli hook di esecuzione di cui è necessario essere a conoscenza durante le operazioni di failover e reverse.

- Durante il failover, gli hook di esecuzione vengono copiati automaticamente dal cluster di origine a quello di destinazione. Non è necessario ricrearli manualmente. Dopo il failover, gli hook di esecuzione sono presenti nell'applicazione e verranno eseguiti durante qualsiasi azione rilevante.
- Durante l'inversione o la risincronizzazione inversa, tutti gli hook di esecuzione esistenti sull'applicazione vengono rimossi. Quando l'applicazione di origine diventa l'applicazione di destinazione, questi hook di esecuzione non sono più validi e vengono eliminati per impedirne l'esecuzione.

Per saperne di più sugli hook di esecuzione, fare riferimento a"Gestire gli hook di esecuzione Trident Protect" .

### Impostare una relazione di replica

L'impostazione di una relazione di replica comporta quanto segue:

- Scegliere la frequenza con cui si desidera che Trident Protect esegua uno snapshot dell'app (che include le risorse Kubernetes dell'app e gli snapshot del volume per ciascuno dei volumi dell'app)
- Scelta del programma di replica (include risorse Kubernetes nonché dati dei volumi persistenti)
- Impostazione dell'ora in cui eseguire l'istantanea

**Fasi**

1. Nel cluster di origine, creare un AppVault per l'applicazione di origine. A seconda del provider di storage, modificare un esempio in "Risorse personalizzate AppVault" per adattare il proprio ambiente:

**Creare un AppVault utilizzando una CR**

a. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-appvault-primary-source.yaml`).

b. Configurare i seguenti attributi:

- **metadata.name**: (*required*) il nome della risorsa personalizzata AppVault. Prendere nota del nome scelto, poiché altri file CR necessari per una relazione di replica fanno riferimento a questo valore.
- **spec.providerConfig**: (*required*) Memorizza la configurazione necessaria per accedere ad AppVault utilizzando il provider specificato. Scegli un bucketName e tutti gli altri dettagli necessari per il tuo provider. Prendere nota dei valori scelti, poiché altri file CR necessari per una relazione di replica fanno riferimento a questi valori. Fare riferimento a "Risorse personalizzate AppVault" per esempi di CRS AppVault con altri provider.
- **spec.providerCredentials**: (*required*) archivia i riferimenti a qualsiasi credenziale richiesta per accedere ad AppVault utilizzando il provider specificato.
  - **spec.providerCredentials.valueFromSecret**: (*required*) indica che il valore della credenziale deve provenire da un segreto.
    - **Key**: (*required*) la chiave valida del segreto da selezionare.
    - **Nome**: (*obbligatorio*) Nome del segreto che contiene il valore per questo campo. Deve trovarsi nello stesso spazio dei nomi.
  - **spec.providerCredentials.secretAccessKey**: (*required*) la chiave di accesso utilizzata per accedere al provider. Il **nome** deve corrispondere a **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*required*) determina cosa fornisce il backup; ad esempio, NetApp ONTAP S3, S3 generico, Google Cloud o Microsoft Azure. Valori possibili:
  - aws
  - azure
  - gcp
  - generico-s3
  - ONTAP-s3
  - StorageGRID-s3

c. Dopo aver popolato il `trident-protect-appvault-primary-source.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n
trident-protect
```

**Creare un AppVault utilizzando la CLI**

a. Creare AppVault, sostituendo i valori tra parentesi con le informazioni dell'ambiente:

```
tridentctl-protect create vault Azure <vault-name> --account
<account-name> --bucket <bucket-name> --secret <secret-name> -n
trident-protect
```

2. Nel cluster di origine, creare l'applicazione di origine CR:

**Creare l'applicazione di origine utilizzando una CR**

a. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-app-source.yaml`).

b. Configurare i seguenti attributi:

- **metadata.name**: (*required*) il nome della risorsa personalizzata dell'applicazione. Prendere nota del nome scelto, poiché altri file CR necessari per una relazione di replica fanno riferimento a questo valore.
- **spec.includedNamespaces**: (*required*) un array di spazi dei nomi e di etichette associate. Utilizzare i nomi degli spazi dei nomi e, facoltativamente, restringere l'ambito degli spazi dei nomi con le etichette per specificare le risorse esistenti negli spazi dei nomi elencati di seguito. Lo spazio dei nomi dell'applicazione deve far parte di questo array.

  **Esempio YAML**:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: my-app-namespace
      labelSelector: {}
```

c. Dopo aver popolato il `trident-protect-app-source.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-namespace
```

**Creare l'applicazione di origine utilizzando l'interfaccia CLI**

a. Creare l'applicazione di origine. Ad esempio:

```
tridentctl-protect create app <my-app-name> --namespaces
<namespaces-to-be-included> -n <my-app-namespace>
```

3. Facoltativamente, sul cluster di origine, eseguire uno snapshot dell'applicazione di origine. Questo snapshot viene utilizzato come base per l'applicazione sul cluster di destinazione. Se si salta questo passaggio, sarà necessario attendere l'esecuzione del prossimo snapshot pianificato per avere uno snapshot recente. Per creare uno snapshot on-demand, fare riferimento a "Crea un'istantanea on-demand".

4. Nel cluster di origine, creare la pianificazione della replica CR:

Oltre alla pianificazione fornita di seguito, si consiglia di creare una pianificazione separata per gli snapshot giornalieri con un periodo di conservazione di 7 giorni per mantenere uno snapshot comune tra i cluster ONTAP peer. Ciò garantisce che gli snapshot siano disponibili fino a 7 giorni, ma il periodo di conservazione può essere personalizzato in base alle esigenze dell'utente.

In caso di failover, il sistema può utilizzare questi snapshot per un massimo di 7 giorni per le operazioni di reverse. Questo approccio rende il processo di reverse più rapido ed efficiente, poiché verranno trasferite solo le modifiche apportate dall'ultimo snapshot, non tutti i dati.

Se una pianificazione esistente per l'applicazione soddisfa già i requisiti di conservazione desiderati, non sono necessarie pianificazioni aggiuntive.

**Creare la pianificazione della replica utilizzando un CR**

a. Creare una pianificazione di replica per l'applicazione di origine:

   i. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-schedule.yaml`).

   ii. Configurare i seguenti attributi:

   - **metadata.name**: (*required*) il nome della risorsa personalizzata di pianificazione.
   - **spec.appVaultRef**: (*Obbligatorio*) Questo valore deve corrispondere al campo metadata.name dell'AppVault per l'applicazione di origine.
   - **spec.applicationRef**: (*Obbligatorio*) Questo valore deve corrispondere al campo metadata.name del CR dell'applicazione di origine.
   - **Spec.backupRetention**: (*required*) questo campo è obbligatorio e il valore deve essere impostato su 0.
   - **Spec.Enabled**: Deve essere impostato su true.
   - **spec.granularity**: deve essere impostato su `Custom`.
   - **Spec.recurrenceRule**: Consente di definire una data di inizio nell'ora UTC e un intervallo di ricorrenza.
   - **Spec.snapshotRetention**: Deve essere impostato su 2.

     Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule
  namespace: my-app-namespace
spec:
  appVaultRef: my-appvault-name
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: Custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

   i. Dopo aver popolato il `trident-protect-schedule.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-
namespace
```

**Creare la pianificazione della replica utilizzando la CLI**

a. Crea la pianificazione della replica, sostituendo i valori tra parentesi con le informazioni provenienti dal tuo ambiente:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule <rule> --snapshot-retention
<snapshot_retention_count> -n <my_app_namespace>
```

**Esempio:**

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule  "DTSTART:20220101T000200Z
\nRRULE:FREQ=MINUTELY;INTERVAL=5" --snapshot-retention 2 -n
<my_app_namespace>
```

5. Nel cluster di destinazione, creare un'applicazione di origine AppVault CR identica a quella AppVault CR applicata al cluster di origine e assegnargli un nome (ad esempio, `trident-protect-appvault-primary-destination.yaml`).
6. Applicare la CR:

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n
trident-protect
```

7. Creare una destinazione AppVault CR per l'applicazione di destinazione sul cluster di destinazione. A seconda del provider di storage, modificare un esempio in "Risorse personalizzate AppVault" per adattare il proprio ambiente:
   a. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-appvault-secondary-destination.yaml`).
   b. Configurare i seguenti attributi:
      - **metadata.name**: (*required*) il nome della risorsa personalizzata AppVault. Prendere nota del nome scelto, poiché altri file CR necessari per una relazione di replica fanno riferimento a questo valore.
      - **spec.providerConfig**: (*required*) Memorizza la configurazione necessaria per accedere ad AppVault utilizzando il provider specificato. Scegliere una `bucketName` e tutte le altre informazioni necessarie per il provider. Prendere nota dei valori scelti, poiché altri file CR necessari per una relazione di replica fanno riferimento a questi valori. Fare riferimento a "Risorse personalizzate AppVault" per esempi di CRS AppVault con altri provider.

- **spec.providerCredentials**: (*required*) archivia i riferimenti a qualsiasi credenziale richiesta per accedere ad AppVault utilizzando il provider specificato.
  - **spec.providerCredentials.valueFromSecret**: (*required*) indica che il valore della credenziale deve provenire da un segreto.
    - **Key**: (*required*) la chiave valida del segreto da selezionare.
    - **Nome**: (*obbligatorio*) Nome del segreto che contiene il valore per questo campo. Deve trovarsi nello stesso spazio dei nomi.
  - **spec.providerCredentials.secretAccessKey**: (*required*) la chiave di accesso utilizzata per accedere al provider. Il **nome** deve corrispondere a **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*required*) determina cosa fornisce il backup; ad esempio, NetApp ONTAP S3, S3 generico, Google Cloud o Microsoft Azure. Valori possibili:
  - aws
  - azure
  - gcp
  - generico-s3
  - ONTAP-s3
  - StorageGRID-s3

c. Dopo aver popolato il `trident-protect-appvault-secondary-destination.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml
-n trident-protect
```

8. Nel cluster di destinazione, creare un file CR AppMirrorRelationship.

> Quando si utilizza una CR, creare manualmente lo spazio dei nomi di destinazione prima di applicare la CR. Trident Protect crea automaticamente gli spazi dei nomi solo quando si utilizza la CLI.

**Creare una relazione AppMirrorRelationship utilizzando una CR**

a. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-relationship.yaml`).

b. Configurare i seguenti attributi:

- **metadata.name:** (obbligatorio) il nome della risorsa personalizzata AppMirrorRelationship.
- **spec.destinationAppVaultRef**: (*required*) questo valore deve corrispondere al nome dell'AppVault per l'applicazione di destinazione sul cluster di destinazione.
- **spec.namespaceMapping**: (*required*) gli spazi dei nomi di destinazione e di origine devono corrispondere allo spazio dei nomi dell'applicazione definito nella rispettiva CR dell'applicazione.
- **Spec.sourceAppVaultRef**: (*required*) questo valore deve corrispondere al nome dell'AppVault per l'applicazione di origine.
- **Spec.sourceApplicationName**: (*required*) questo valore deve corrispondere al nome dell'applicazione di origine definita nell'applicazione di origine CR.
- **spec.sourceApplicationUID**: (obbligatorio) Questo valore deve corrispondere all'UID dell'applicazione sorgente definita nel CR dell'applicazione sorgente.
- **spec.storageClassName**: (*Facoltativo*) Scegli il nome di una classe di archiviazione valida sul cluster. La classe di archiviazione deve essere collegata a una VM di archiviazione ONTAP collegata in peering con l'ambiente di origine. Se non viene specificata la classe di archiviazione, verrà utilizzata per impostazione predefinita la classe di archiviazione predefinita sul cluster.
- **Spec.recurrenceRule**: Consente di definire una data di inizio nell'ora UTC e un intervallo di ricorrenza.

  Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: amr-16061e80-1b05-4e80-9d26-d326dc1953d8
  namespace: my-app-namespace
spec:
  desiredState: Established
  destinationAppVaultRef: generic-s3-trident-protect-dst-bucket-
8fe0b902-f369-4317-93d1-ad7f2edc02b5
  namespaceMapping:
    - destination: my-app-namespace
      source: my-app-namespace
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  sourceAppVaultRef: generic-s3-trident-protect-src-bucket-
b643cc50-0429-4ad5-971f-ac4a83621922
  sourceApplicationName: my-app-name
  sourceApplicationUID: 7498d32c-328e-4ddd-9029-122540866aeb
  storageClassName: sc-vsim-2
```

c. Dopo aver popolato il `trident-protect-relationship.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-
namespace
```

**Creare un AppMirrorRelationship utilizzando l'interfaccia CLI**

a. Crea e applica l'oggetto AppMirrorRelationship, sostituendo i valori tra parentesi con le informazioni provenienti dal tuo ambiente:

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirorrelationship> --destination-app-vault
<my_vault_name> --source-app-vault <my_vault_name> --recurrence
-rule <rule> --namespace-mapping <ns_mapping> --source-app-id
<source_app_UID> --source-app <my_source_app_name> --storage
-class <storage_class_name> -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. (*Optional*) nel cluster di destinazione, verificare lo stato e lo stato della relazione di replica:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

#### Failover sul cluster di destinazione

Utilizzando Trident Protect è possibile eseguire il failover delle applicazioni replicate su un cluster di destinazione. Questa procedura interrompe la relazione di replica e porta l'app online sul cluster di destinazione. Trident Protect non arresta l'app sul cluster di origine se era operativa.

**Fasi**

1. Nel cluster di destinazione, modificare il file CR AppMirrorRelationship (ad esempio, `trident-protect-relationship.yaml`) e modificare il valore di **spec.desiredState** in `Promoted`.
2. Salvare il file CR.
3. Applicare la CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. (*Optional*) creare tutte le pianificazioni di protezione necessarie per l'applicazione in cui è stato eseguito il failover.
5. (*Optional*) controllare lo stato e lo stato della relazione di replica:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

#### Risincronizzazione di una relazione di replica non riuscita

L'operazione di risincronizzazione ristabilisce la relazione di replica. Dopo aver eseguito un'operazione di risincronizzazione, l'applicazione di origine diventa l'applicazione in esecuzione e tutte le modifiche apportate all'applicazione in esecuzione sul cluster di destinazione vengono scartate.

Il processo arresta l'applicazione sul cluster di destinazione prima di ristabilire la replica.

Tutti i dati scritti nell'applicazione di destinazione durante il failover andranno persi.

**Fasi**

1. Opzionale: Nel cluster di origine, creare uno snapshot dell'applicazione di origine. In questo modo si garantisce che vengano acquisite le ultime modifiche dal cluster di origine.
2. Nel cluster di destinazione, modificare il file CR AppMirrorRelationship (ad esempio, `trident-protect-relationship.yaml`) e modificare il valore di spec.desiredState in `Established`.
3. Salvare il file CR.
4. Applicare la CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. Rimuovere eventuali pianificazioni di protezione sul cluster di destinazione per proteggere l'applicazione in cui è stato eseguito il failover. Qualsiasi pianificazione rimanente causa errori di snapshot dei volumi.

#### Risincronizzazione inversa di una relazione di replica non riuscita

Quando si esegue la risincronizzazione inversa di una relazione di replica non riuscita, l'applicazione di destinazione diventa l'applicazione di origine e l'origine diventa la destinazione. Le modifiche apportate all'applicazione di destinazione durante il failover vengono mantenute.

**Fasi**

1. Nel cluster di destinazione originale, eliminare la CR AppMirrorRelationship. Ciò fa sì che la destinazione diventi l'origine. Rimuovere eventuali pianificazioni relative alla protezione sul nuovo cluster di destinazione.
2. Impostare una relazione di replica applicando i file CR utilizzati originariamente per impostare la relazione con i cluster opposti.
3. Assicurarsi che la nuova destinazione (cluster di origine originale) sia configurata con entrambi i CRS AppVault.
4. Impostare una relazione di replica sul cluster opposto, configurando i valori per la direzione inversa.

### Invertire la direzione di replica dell'applicazione

Quando si inverte la direzione della replica, Trident Protect sposta l'applicazione sul backend di archiviazione di destinazione, continuando a replicare sul backend di archiviazione di origine. Trident Protect arresta l'applicazione di origine e replica i dati nella destinazione prima di eseguire il failover sull'app di destinazione.

In questa situazione, si sta sostituendo l'origine e la destinazione.

**Fasi**

1. Nel cluster di origine, creare uno snapshot di arresto:

**Creare un'istantanea di arresto utilizzando una CR**

a. Disattivare le pianificazioni dei criteri di protezione per l'applicazione di origine.

b. Creare un file ShutdownSnapshot CR:

i. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-shutdownsnapshot.yaml`).

ii. Configurare i seguenti attributi:

- **metadata.name**: (*required*) il nome della risorsa personalizzata.
- **Spec.AppVaultRef**: (*required*) questo valore deve corrispondere al campo metadata.name dell'AppVault per l'applicazione di origine.
- **Spec.ApplicationRef**: (*required*) questo valore deve corrispondere al campo metadata.name del file CR dell'applicazione di origine.

Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ShutdownSnapshot
metadata:
  name: replication-shutdown-snapshot-afc4c564-e700-4b72-86c3-
c08a5dbe844e
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  applicationRef: my-app-name
```

c. Dopo aver popolato il `trident-protect-shutdownsnapshot.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-
namespace
```

**Creare uno snapshot di arresto utilizzando l'interfaccia CLI**

a. Creare l'istantanea di arresto, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente. Ad esempio:

```
tridentctl-protect create shutdownsnapshot <my_shutdown_snapshot>
--appvault <my_vault> --app <app_to_snapshot> -n
<application_namespace>
```

2. Sul cluster di origine, dopo il completamento dello snapshot di arresto, ottenere lo stato dello snapshot di arresto:

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. Nel cluster di origine, trovare il valore di **shutdownsnapshot.status.appArchivePath** utilizzando il seguente comando e registrare l'ultima parte del percorso del file (chiamato anche nome di base; questo sarà tutto dopo l'ultima barra):

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o
jsonpath='{.status.appArchivePath}'
```

4. Eseguire un failover dal nuovo cluster di destinazione al nuovo cluster di origine, con la seguente modifica:

> Nel passaggio 2 della procedura di failover, includere il `spec.promotedSnapshot` campo nel file CR AppMirrorRelationship e impostarne il valore sul nome di base registrato nel passaggio 3 di cui sopra.

5. Eseguire le operazioni di risincronizzazione inversa descritte in Risincronizzazione inversa di una relazione di replica non riuscita.
6. Attiva le pianificazioni della protezione sul nuovo cluster di origine.

#### Risultato

A causa della replica inversa, si verificano le seguenti azioni:

- Viene acquisita un'istantanea delle risorse Kubernetes dell'applicazione di origine.
- I pod dell'applicazione di origine vengono interrotti correttamente eliminando le risorse Kubernetes dell'applicazione (lasciando PVC e PVS in posizione).
- Una volta spenti i pod, vengono acquisite e replicate le istantanee dei volumi dell'applicazione.
- Le relazioni di SnapMirror vengono interrotte, rendendo i volumi di destinazione pronti per la lettura/scrittura.
- Le risorse Kubernetes dell'applicazione vengono ripristinate dallo snapshot pre-shutdown, utilizzando i dati del volume replicati dopo l'arresto dell'applicazione di origine.
- La replica viene ristabilita in senso inverso.

#### Eseguire il failback delle applicazioni nel cluster di origine originale

Utilizzando Trident Protect, è possibile ottenere il "fail back" dopo un'operazione di failover utilizzando la seguente sequenza di operazioni. In questo flusso di lavoro per ripristinare la direzione di replicazione originale, Trident Protect replica (risincronizza) tutte le modifiche apportate all'applicazione di origine prima di invertire la direzione di replicazione.

Questo processo inizia da una relazione che ha completato un failover verso una destinazione e prevede i seguenti passaggi:

- Iniziare con uno stato di failover.
- Risincronizzazione inversa della relazione di replica.

> Non eseguire una normale operazione di risincronizzazione, in quanto i dati scritti nel cluster di destinazione verranno eliminati durante la procedura di failover.

- Invertire la direzione di replica.

**Fasi**

1. Eseguire i Risincronizzazione inversa di una relazione di replica non riuscitapassaggi.
2. Eseguire i Invertire la direzione di replica dell'applicazionepassaggi.

#### Eliminare una relazione di replica

È possibile eliminare una relazione di replica in qualsiasi momento. Quando si elimina la relazione di replica dell'applicazione, vengono generate due applicazioni separate senza alcuna relazione tra di esse.

**Fasi**

1. Nel cluster di desinazione corrente, eliminare AppMirrorRelationship CR:

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

## Migrare le applicazioni utilizzando Trident Protect

È possibile migrare le applicazioni tra cluster o in classi di archiviazione diverse ripristinando i dati di backup.

> Quando si esegue la migrazione di un'applicazione, tutti i collegamenti di esecuzione configurati per l'applicazione vengono migrati con l'applicazione. Se è presente un gancio di esecuzione post-ripristino, viene eseguito automaticamente come parte dell'operazione di ripristino.

### Operazioni di backup e ripristino

Per eseguire operazioni di backup e ripristino per i seguenti scenari, è possibile automatizzare attività di backup e ripristino specifiche.

#### Clona nello stesso cluster

Per clonare un'applicazione nello stesso cluster, crea una snapshot o un backup e ripristina i dati nello stesso cluster.

**Fasi**

1. Effettuare una delle seguenti operazioni:
   a. "Creare un'istantanea".
   b. "Creare un backup".
2. Nello stesso cluster, eseguire una delle seguenti operazioni, a seconda che sia stato creato uno snapshot o un backup:

a. "Ripristinare i dati dalla snapshot".

b. "Ripristinare i dati dal backup".

#### Clona in un cluster diverso

Per clonare un'applicazione su un cluster diverso (eseguire un clone tra cluster), creare un backup sul cluster di origine, quindi ripristinare il backup su un cluster diverso. Assicurarsi che Trident Protect sia installato sul cluster di destinazione.

È possibile replicare un'applicazione tra cluster diversi utilizzando "Replica SnapMirror".

**Fasi**

1. "Creare un backup".
2. Verificare che AppVault CR per il bucket di storage a oggetti che contiene il backup sia stato configurato sul cluster di destinazione.
3. Sul cluster di destinazione, "ripristinare i dati dal backup".

### Eseguire la migrazione delle applicazioni da una classe di storage a un'altra

È possibile migrare le applicazioni da una classe di archiviazione a una diversa ripristinando un backup nella classe di archiviazione di destinazione.

Ad esempio (escludendo i segreti dalla CR di ripristino):

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: "${snapshotRestoreCRName}"
spec:
  appArchivePath: "${snapshotArchivePath}"
  appVaultRef: "${appVaultCRName}"
  namespaceMapping:
    - destination: "${destinationNamespace}"
      source: "${sourceNamespace}"
  storageClassMapping:
    - destination: "${destinationStorageClass}"
      source: "${sourceStorageClass}"
  resourceFilter:
    resourceMatchers:
      kind: Secret
      version: v1
    resourceSelectionCriteria: exclude
```

**Ripristinare l'istantanea utilizzando una CR**

**Fasi**

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-snapshot-restore-cr.yaml`.
2. Nel file creato, configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono memorizzati i contenuti dello snapshot. Per trovare il percorso, utilizzare il seguente comando:

     ```
     kubectl get snapshots <my-snapshot-name> -n trident-protect -o jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui sono memorizzati i contenuti dello snapshot.
   - **spec.namespaceMapping**: mappatura dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituire `my-source-namespace` e `my-destination-namespace` con le informazioni del proprio ambiente.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: trident-protect
     spec:
       appArchivePath: my-snapshot-path
       appVaultRef: appvault-name
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. Se si desidera, è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda le risorse contrassegnate con determinate etichette:
   - **ResourceFilter.resourceSelectionCriteria**: (Necessario per il filtraggio) utilizzare `include or exclude` per includere o escludere una risorsa definita in resourceMatcher. Aggiungere i seguenti parametri resourceMatcher per definire le risorse da includere o escludere:
     - **ResourceFilter.resourceMatchers**: Una matrice di oggetti resourceMatcher. Se si definiscono più elementi in questa matrice, questi corrispondono come un'operazione OR e i campi all'interno di ogni elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.
       - **ResourceMatchers[].group**: (*Optional*) Gruppo della risorsa da filtrare.
       - **ResourceMatchers[].Kind**: (*Optional*) tipo di risorsa da filtrare.

- **ResourceMatchers[].version**: (*Optional*) versione della risorsa da filtrare.
- **ResourceMatchers[].names**: (*Optional*) nomi nel campo Kubernetes metadata.name della risorsa da filtrare.
- **ResourceMatchers[].namespaces**: (*Optional*) Namespaces nel campo Kubernetes metadata.name della risorsa da filtrare.
- **ResourceMatchers[].labelSelectors**: (*Optional*) stringa del selettore di etichette nel campo Kubernetes metadata.name della risorsa come definito nella "Documentazione Kubernetes" . Ad esempio: `"trident.netapp.io/os=linux"`.

  Ad esempio:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Dopo aver popolato il `trident-protect-snapshot-restore-cr.yaml` file con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

**Ripristinare la snapshot utilizzando la CLI**

**Fasi**

1. Ripristinare lo snapshot in uno spazio dei nomi diverso, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente.

   - L'`snapshot`argomento utilizza uno spazio dei nomi e un nome snapshot nel formato `<namespace>/<name>.
   - L'`namespace-mapping`argomento utilizza spazi dei nomi separati da due punti per mappare gli spazi dei nomi di origine agli spazi dei nomi di destinazione corretti nel formato `source1:dest1,source2:dest2.

     Ad esempio:

```
tridentctl-protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

## Gestire gli hook di esecuzione Trident Protect

Un gancio di esecuzione è un'azione personalizzata che è possibile configurare per l'esecuzione in combinazione con un'operazione di protezione dei dati di un'applicazione gestita. Ad esempio, se si dispone di un'applicazione di database, è possibile utilizzare un gancio di esecuzione per mettere in pausa tutte le transazioni del database prima di uno snapshot e riprendere le transazioni al termine dello snapshot. Ciò garantisce snapshot coerenti con l'applicazione.

### Tipi di hook di esecuzione

Trident Protect supporta i seguenti tipi di hook di esecuzione, in base al momento in cui possono essere eseguiti:

- Pre-snapshot
- Post-snapshot
- Pre-backup
- Post-backup
- Post-ripristino
- Post-failover

#### Ordine di esecuzione

Quando viene eseguita un'operazione di protezione dei dati, gli eventi hook di esecuzione hanno luogo nel seguente ordine:

1. Gli eventuali hook di esecuzione pre-operation personalizzati applicabili vengono eseguiti sui container appropriati. È possibile creare ed eseguire tutti gli hook pre-operation personalizzati necessari, ma l'ordine di esecuzione di questi hook prima dell'operazione non è garantito né configurabile.
2. Se applicabile, si verificano blocchi del file system. "Scopri di più sulla configurazione del congelamento del file system con Trident Protect".
3. Viene eseguita l'operazione di protezione dei dati.
4. I filesystem congelati vengono scongelati, se applicabile.
5. Gli eventuali hook di esecuzione post-operation personalizzati applicabili vengono eseguiti sui container appropriati. È possibile creare ed eseguire tutti gli hook post-operation personalizzati necessari, ma l'ordine di esecuzione di questi hook dopo l'operazione non è garantito né configurabile.

Se si creano più hook di esecuzione dello stesso tipo (ad esempio, pre-snapshot), l'ordine di esecuzione di tali hook non è garantito. Tuttavia, è garantito l'ordine di esecuzione di ganci di tipi diversi. Ad esempio, di seguito è riportato l'ordine di esecuzione di una configurazione che ha tutti i diversi tipi di ganci:

1. Hook pre-snapshot eseguiti
2. Esecuzione di hook post-snapshot
3. Hook pre-backup eseguiti
4. Hook post-backup eseguiti

L'esempio dell'ordine precedente si applica solo quando si esegue un backup che non utilizza uno snapshot esistente.

Prima di abilitarli in un ambiente di produzione, è necessario verificare sempre gli script hook di esecuzione. È possibile utilizzare il comando 'kubectl exec' per testare comodamente gli script. Dopo aver attivato gli hook di esecuzione in un ambiente di produzione, testare le snapshot e i backup risultanti per assicurarsi che siano coerenti. Per eseguire questa operazione, clonare l'applicazione in uno spazio dei nomi temporaneo, ripristinare lo snapshot o il backup e quindi testare l'applicazione.

Se un gancio di esecuzione pre-snapshot aggiunge, modifica o rimuove le risorse Kubernetes, queste modifiche sono incluse nella snapshot o nel backup e in qualsiasi operazione di ripristino successiva.

**Note importanti sugli hook di esecuzione personalizzati**

Quando si pianificano gli hook di esecuzione per le applicazioni, considerare quanto segue.

- Un gancio di esecuzione deve utilizzare uno script per eseguire le azioni. Molti hook di esecuzione possono fare riferimento allo stesso script.
- Trident Protect richiede che gli script utilizzati dagli hook di esecuzione siano scritti nel formato degli script shell eseguibili.
- La dimensione dello script è limitata a 96 KB.
- Trident Protect utilizza le impostazioni dell'hook di esecuzione e tutti i criteri corrispondenti per determinare quali hook sono applicabili a un'operazione di snapshot, backup o ripristino.

Poiché gli hook di esecuzione spesso riducono o disattivano completamente le funzionalità dell'applicazione con cui vengono eseguiti, si consiglia di ridurre al minimo il tempo necessario per l'esecuzione degli hook di esecuzione personalizzati. Se si avvia un'operazione di backup o snapshot con gli hook di esecuzione associati, ma poi si annulla, gli hook possono ancora essere eseguiti se l'operazione di backup o snapshot è già iniziata. Ciò significa che la logica utilizzata in un gancio di esecuzione post-backup non può presumere che il backup sia stato completato.

**Esecuzione dei filtri hook**

Quando si aggiunge o si modifica un gancio di esecuzione per un'applicazione, è possibile aggiungere filtri al gancio di esecuzione per gestire i contenitori corrispondenti. I filtri sono utili per le applicazioni che utilizzano la stessa immagine container su tutti i container, ma possono utilizzare ogni immagine per uno scopo diverso (ad esempio Elasticsearch). I filtri consentono di creare scenari in cui gli hook di esecuzione vengono eseguiti su alcuni container identici, ma non necessariamente su tutti. Se si creano più filtri per un singolo gancio di esecuzione, questi vengono combinati con un operatore AND logico. È possibile avere fino a 10 filtri attivi per gancio di esecuzione.

Ogni filtro aggiunto a un hook di esecuzione utilizza un'espressione regolare per abbinare i contenitori nel cluster. Quando un hook corrisponde a un contenitore, eseguirà lo script associato su quel contenitore. Le espressioni regolari per i filtri utilizzano la sintassi Regular Expression 2 (RE2), che non supporta la creazione di un filtro che escluda i contenitori dall'elenco delle corrispondenze. Per informazioni sulla sintassi supportata da Trident Protect per le espressioni regolari nei filtri di hook di esecuzione, vedere "Supporto della sintassi RE2 (Regular Expression 2)" .

> Se si aggiunge un filtro dello spazio dei nomi a un gancio di esecuzione che viene eseguito dopo un'operazione di ripristino o clonazione e l'origine e la destinazione del ripristino o del clone si trovano in spazi dei nomi diversi, il filtro dello spazio dei nomi viene applicato solo allo spazio dei nomi di destinazione.

### Esempi di gancio di esecuzione

Visita il sito "Progetto NetApp Verda GitHub" per scaricare i veri hook di esecuzione per le app più diffuse, come Apache Cassandra ed Elasticsearch. Puoi anche vedere esempi e trovare idee per strutturare i tuoi hook di esecuzione personalizzati.

### Creare un gancio di esecuzione

È possibile creare un hook di esecuzione personalizzato per un'app utilizzando Trident Protect. Per creare hook di esecuzione è necessario disporre delle autorizzazioni di Proprietario, Amministratore o Membro.

**Utilizzare un CR**

**Fasi**

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-hook.yaml`.
2. Configurare i seguenti attributi in modo che corrispondano all'ambiente Trident Protect e alla configurazione del cluster:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.applicationRef**: (*required*) il nome Kubernetes dell'applicazione per la quale eseguire l'hook di esecuzione.
   - **Spec.stage**: (*required*) stringa che indica quale fase durante l'azione deve essere eseguita l'hook di esecuzione. Valori possibili:
     - Pre
     - Post
   - **Spec.action**: (*required*) stringa che indica l'azione che verrà eseguita dall'hook di esecuzione, presupponendo che tutti i filtri di hook di esecuzione specificati siano corrispondenti. Valori possibili:
     - Snapshot
     - Backup
     - Ripristinare
     - Failover
   - **Spec.Enabled**: (*Optional*) indica se questo gancio di esecuzione è abilitato o disabilitato. Se non specificato, il valore predefinito è true.
   - **Spec.hookSource**: (*required*) stringa contenente lo script hook codificato in base64.
   - **Spec.timeout**: (*Optional*) Un numero che definisce il tempo in minuti per il quale il gancio di esecuzione può essere eseguito. Il valore minimo è 1 minuto e, se non specificato, il valore predefinito è 25 minuti.
   - **Spec.arguments**: (*Optional*) elenco YAML di argomenti che è possibile specificare per l'hook di esecuzione.
   - **Spec.matchingCriteria**: (*Optional*) un elenco facoltativo di coppie di valori chiave di criteri, ciascuna coppia costituendo un filtro di hook di esecuzione. È possibile aggiungere fino a 10 filtri per ogni collegamento di esecuzione.
   - **Spec.matchingCriteria.type**: (*Optional*) Una stringa che identifica il tipo di filtro del gancio di esecuzione. Valori possibili:
     - Immagine containerImage
     - ContainerName
     - PodName
     - PodLabel
     - NamespaceName
   - **Spec.matchingCriteria.value**: (*Optional*) Una stringa o Un'espressione regolare che identifica il valore del filtro dell'hook di esecuzione.

     Esempio YAML:

```
apiVersion: protect.trident.netapp.io/v1
kind: ExecHook
metadata:
  name: example-hook-cr
  namespace: my-app-namespace
  annotations:
    astra.netapp.io/astra-control-hook-source-id:
/account/test/hookSource/id
spec:
  applicationRef: my-app-name
  stage: Pre
  action: Snapshot
  enabled: true
  hookSource: IyEvYmluL2Jhc2gKZWNobyAiZXhhbXBsZSBzY3JpcHQiCg==
  timeout: 10
  arguments:
    - FirstExampleArg
    - SecondExampleArg
  matchingCriteria:
    - type: containerName
      value: mysql
    - type: containerImage
      value: bitnami/mysql
    - type: podName
      value: mysql
    - type: namespaceName
      value: mysql-a
    - type: podLabel
      value: app.kubernetes.io/component=primary
    - type: podLabel
      value: helm.sh/chart=mysql-10.1.0
    - type: podLabel
      value: deployment-type=production
```

3. Dopo aver popolato il file CR con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-hook.yaml
```

**Utilizzare la CLI**

**Fasi**

1. Creare il gancio di esecuzione, sostituendo i valori tra parentesi con le informazioni dell'ambiente. Ad esempio:

```
tridentctl-protect create exechook <my_exec_hook_name> --action
<action_type> --app <app_to_use_hook> --stage <pre_or_post_stage>
--source-file <script-file> -n <application_namespace>
```

**Eseguire manualmente un gancio di esecuzione**

È possibile eseguire manualmente un gancio di esecuzione a scopo di test o se è necessario eseguire nuovamente il gancio manualmente dopo un errore. È necessario disporre delle autorizzazioni Proprietario, Amministratore o membro per eseguire manualmente i ganci di esecuzione.

L'esecuzione manuale di un gancio di esecuzione consiste in due passaggi di base:

1. Creare un backup delle risorse, che raccoglie le risorse e ne crea un backup, determinando dove verrà eseguito il hook
2. Eseguire il gancio di esecuzione sul backup

### Passaggio 1: Creare un backup delle risorse

**Utilizzare un CR**

**Fasi**

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-resource-backup.yaml`.
2. Configurare i seguenti attributi in modo che corrispondano all'ambiente Trident Protect e alla configurazione del cluster:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.applicationRef**: (*required*) il nome Kubernetes dell'applicazione per cui creare il backup delle risorse.
   - **Spec.appVaultRef**: (*required*) il nome dell'AppVault in cui sono memorizzati i contenuti di backup.
   - **Spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono memorizzati i contenuti di backup. Per trovare il percorso, utilizzare il seguente comando:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

     Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ResourceBackup
metadata:
  name: example-resource-backup
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
```

3. Dopo aver popolato il file CR con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-resource-backup.yaml
```

**Utilizzare la CLI**

**Fasi**

1. Creare il backup, sostituendo i valori tra parentesi con le informazioni provenienti dall'ambiente. Ad esempio:

```
tridentctl protect create resourcebackup <my_backup_name> --app
<my_app_name> --appvault <my_appvault_name> -n
<my_app_namespace> --app-archive-path <app_archive_path>
```

2. Visualizzare lo stato del backup. È possibile utilizzare questo comando di esempio ripetutamente fino al completamento dell'operazione:

```
tridentctl protect get resourcebackup -n <my_app_namespace>
<my_backup_name>
```

3. Verificare che il backup sia stato eseguito correttamente:

```
kubectl describe resourcebackup <my_backup_name>
```

### Fase 2: Eseguire il gancio di esecuzione

**Utilizzare un CR**

**Fasi**

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-hook-run.yaml`.
2. Configurare i seguenti attributi in modo che corrispondano all'ambiente Trident Protect e alla configurazione del cluster:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.applicationRef**: (*required*) assicurarsi che questo valore corrisponda al nome dell'applicazione dal ResourceBackup CR creato nel passaggio 1.
   - **Spec.appVaultRef**: (*required*) assicurarsi che questo valore corrisponda all'appVaultRef del ResourceBackup CR creato nel passaggio 1.
   - **Spec.appArchivePath**: Assicurarsi che questo valore corrisponda all'appArchivePath del ResourceBackup CR creato nel passaggio 1.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.action**: (*required*) stringa che indica l'azione che verrà eseguita dall'hook di esecuzione, presupponendo che tutti i filtri di hook di esecuzione specificati siano corrispondenti. Valori possibili:
     - Snapshot
     - Backup
     - Ripristinare
     - Failover
   - **Spec.stage**: (*required*) stringa che indica quale fase durante l'azione deve essere eseguita l'hook di esecuzione. Questa corsa del gancio non farà funzionare i ganci in nessun altro stadio. Valori possibili:
     - Pre
     - Post

     Esempio YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ExecHooksRun
metadata:
  name: example-hook-run
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
  stage: Post
  action: Failover
```

3. Dopo aver popolato il file CR con i valori corretti, applicare la CR:

```
kubectl apply -f trident-protect-hook-run.yaml
```

**Utilizzare la CLI**

**Fasi**

1. Creare la richiesta di esecuzione hook manuale:

```
tridentctl protect create exechooksrun <my_exec_hook_run_name>
-n <my_app_namespace> --action snapshot --stage <pre_or_post>
--app <my_app_name> --appvault <my_appvault_name> --path
<my_backup_name>
```

2. Controllare lo stato della sequenza di aggancio esecuzione. È possibile eseguire questo comando ripetutamente fino al completamento dell'operazione:

```
tridentctl protect get exechooksrun -n <my_app_namespace>
<my_exec_hook_run_name>
```

3. Descrivere l'oggetto exechooksrun per visualizzare i dettagli e lo stato finali:

```
kubectl -n <my_app_namespace> describe exechooksrun
<my_exec_hook_run_name>
```

# Disinstallare Trident Protect

Potrebbe essere necessario rimuovere i componenti Trident Protect se si esegue l'aggiornamento da una versione di prova a una versione completa del prodotto.

Per rimuovere Trident Protect, procedere come segue.

**Fasi**

1. Rimuovere i file CR Trident Protect:

   Questo passaggio non è necessario per la versione 25.06 e successive.

   ```
   helm uninstall -n trident-protect trident-protect-crds
   ```

2. Rimuovere Trident Protect:

   ```
   helm uninstall -n trident-protect trident-protect
   ```

3. Rimuovere lo spazio dei nomi Trident Protect:

   ```
   kubectl delete ns trident-protect
   ```