Proteggi le applicazioni con Trident Protect

Trident

NetApp
September 26, 2025

Sommario

Proteggi le applicazioni con Trident Protect . . . 1
- Informazioni su Trident Protect . . . 1
  - Quali sono le prossime novità? . . . 1
- Installare Trident Protect . . . 1
  - Requisiti di Trident Protect . . . 1
  - Installare e configurare Trident Protect . . . 4
  - Installare il plugin Trident Protect CLI . . . 10
- Gestire Trident Protect . . . 14
  - Gestire le autorizzazioni e il controllo degli accessi Trident Protect . . . 14
  - Generare un bundle di supporto Trident Protect . . . 20
  - Aggiornare Trident Protect . . . 22
- Gestisci e proteggi le applicazioni . . . 22
  - Utilizzare gli oggetti Trident Protect AppVault per gestire i bucket . . . 22
  - Definire un'applicazione da gestire con Trident Protect . . . 30
  - Proteggi le applicazioni con Trident Protect . . . 32
  - Ripristina le applicazioni utilizzando Trident Protect . . . 40
  - Replica le applicazioni utilizzando NetApp SnapMirror e Trident Protect . . . 56
  - Migrazione delle applicazioni con Trident Protect . . . 69
  - Gestire i hook di esecuzione Trident Protect . . . 73
- Disinstallare Trident Protect . . . 78

Proteggi le applicazioni con Trident Protect

Informazioni su Trident Protect

NetApp Trident Protect offre capacità avanzate di gestione dei dati delle applicazioni che migliorano la funzionalità e la disponibilità delle applicazioni stateful Kubernetes supportate dai sistemi storage NetApp ONTAP e dal provisioner dello storage NetApp Trident CSI. Trident Protect semplifica la gestione, la protezione e lo spostamento dei workload in container nei cloud pubblici e negli ambienti on-premise. Offre anche funzionalità di automazione tramite il proprio API e CLI.

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

Quali sono le prossime novità?

Informazioni sui requisiti di Trident Protect prima di procedere all'installazione:

- "Requisiti di Trident Protect"

Installare Trident Protect

Requisiti di Trident Protect

Inizia subito con la verifica della prontezza del tuo ambiente operativo, dei cluster di applicazioni, delle applicazioni e delle licenze. Assicurati che il tuo ambiente soddisfi questi requisiti per l'implementazione e l'utilizzo di Trident Protect.

Trident protegge la compatibilità del cluster Kubernetes

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

> Assicurarsi che il cluster su cui si installa Trident Protect sia configurato con un controller snapshot in esecuzione e i CRD correlati. Per installare un'unità di controllo istantanee, fare riferimento alla "queste istruzioni".

Trident protegge la compatibilità del backend di storage

Trident Protect supporta i seguenti backend di storage:

- 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:

- Assicurati che lo storage NetApp connesso al cluster stia utilizzando Astra Trident 24,02 o versione successiva (si consiglia Trident 24,10).
  - Se Astra Trident è precedente alla versione 24.06.1 e intendi utilizzare la funzionalità di disaster recovery di NetApp SnapMirror, devi attivare manualmente Astra Control Provisioner.
- Assicurati di avere l'ultima versione di Astra Control Provisioner (installata e abilitata per impostazione predefinita a partire da Astra Trident 24.06.1).
- 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.
- Creare spazi dei nomi delle applicazioni che si intende utilizzare per applicazioni o operazioni di gestione dei dati delle applicazioni. Trident Protect non crea questi spazi dei nomi per l'utente; se si specifica uno spazio dei nomi inesistente in una risorsa personalizzata, l'operazione non verrà eseguita correttamente.

Per i volumi nas-Economy

Trident Protect supporta le operazioni di backup e ripristino su volumi nas-Economy. Al momento snapshot, cloni e replica SnapMirror sui volumi nas-Economy non sono supportati. È necessario abilitare una directory di snapshot per ogni volume economico nas 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 24,10 e 24.10.1 e versioni successive garantiscono comportamenti diversi per la protezione delle applicazioni in esecuzione sulle macchine virtuali KubeVirt. Per entrambe le versioni, è possibile attivare o disattivare il blocco e lo sblocco del file system durante le operazioni di protezione dei dati.

> Per tutte le versioni di Trident Protect, per abilitare o disabilitare la funzionalità di blocco automatico negli ambienti OpenShift, potrebbe essere necessario concedere autorizzazioni privilegiate allo spazio dei nomi delle applicazioni. Ad esempio:
>
> ```
> oc adm policy add-scc-to-user privileged -z default -n
> <application-namespace>
> ```

Trident Protect 24,10

Trident Protect 24,10 non garantisce automaticamente uno stato coerente dei file system delle macchine virtuali KubeVirt durante le operazioni di protezione dei dati. Per proteggere i dati delle macchine virtuali KubeVirt utilizzando Trident Protect 24,10, è necessario abilitare manualmente la funzionalità di blocco/sblocco dei file system prima dell'operazione di protezione dei dati. Ciò garantisce che i filesystem siano in uno stato coerente.

È possibile configurare Trident Protect 24,10 per gestire il blocco e lo sblocco del file system della VM durante le operazioni di protezione dei dati "configurazione della virtualizzazione"utilizzando il seguente comando:

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

Trident Protect 24.10.1 e versioni successive

A partire da Trident Protect 24.10.1, Trident Protect blocca e sblocca automaticamente i file system KubeVirt durante le operazioni di data Protection. 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
```

Requisiti per la replica SnapMirror

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

- NetApp ASA
- NetApp AFF
- NetApp FAS
- 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:

- **Astra Control Provisioner o Trident**: Astra Control Provisioner o Trident deve esistere sia sui cluster Kubernetes di origine che di destinazione che utilizzano ONTAP come backend. Trident Protect supporta la replica con la tecnologia NetApp SnapMirror utilizzando classi di storage supportate dai seguenti driver:
  - `ontap-nas`
  - `ontap-san`
- **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.

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.

- **Astra Control Provisioner o Trident e SVM**: Le SVM remote in fase di migrazione devono essere disponibili per Astra Control Provisioner o Trident nel cluster di destinazione.
- **Backend gestiti**: È necessario aggiungere e gestire i backend di storage ONTAP in Trident Protect per creare una relazione di replica.
- **NVMe over TCP**: Trident Protect non supporta la replica NetApp SnapMirror per backend di storage che utilizzano il protocollo NVMe over TCP.

Configurazione Trident / ONTAP per la replica SnapMirror

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

Installare e configurare Trident Protect

Se l'ambiente in uso soddisfa i requisiti di Trident Protect, è possibile seguire questa procedura per installare Trident Protect sul cluster. È possibile ottenere Trident Protect da NetApp o installarlo dal proprio registro privato. L'installazione da un registro privato è utile se il cluster non riesce ad accedere a Internet.

Per impostazione predefinita, Trident Protect raccoglie le informazioni di supporto utili per qualsiasi caso di supporto NetApp che potrebbe essere aperto, inclusi log, metriche e informazioni sulla topologia di cluster e applicazioni gestite. Trident Protect invia questi bundle di supporto a NetApp secondo una pianificazione giornaliera. Se lo si desidera, è possibile disattivare questa raccolta bundle di supporto quando si installa Trident Protect. È possibile eseguire manualmente "generare un bundle di supporto"in qualsiasi momento.

Installare Trident Protect

Installare Trident Protect di NetApp

Fasi

1. Aggiungere il repository Trident Helm:

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

2. Installare i CRD Trident Protect:

```
helm install trident-protect-crds netapp-trident-protect/trident-
protect-crds --version 100.2410.1 --create-namespace --namespace
trident-protect
```

3. Utilizzare Helm per installare Trident Protect utilizzando uno dei seguenti comandi. Sostituire `<name_of_cluster>` con un nome cluster, che verrà assegnato al cluster e utilizzato per identificare i backup e gli snapshot del cluster:
   - Installare Trident Protect normalmente:

```
helm install trident-protect netapp-trident-protect/trident-
protect --set clusterName=<name_of_cluster> --version 100.2410.1
--create-namespace --namespace trident-protect
```

   - Installare Trident Protect e disattivare i caricamenti giornalieri programmati del pacchetto di supporto Trident Protect AutoSupport:

```
helm install trident-protect netapp-trident-protect/trident-
protect --set autoSupport.enabled=false --set
clusterName=<name_of_cluster> --version 100.2410.1 --create
-namespace --namespace trident-protect
```

Installare Trident Protect da un registro privato

È possibile installare Trident Protect da un registro di immagine privata se il cluster Kubernetes non è in grado di accedere a Internet. In questi esempi, sostituire i valori tra parentesi con le informazioni dell'ambiente:

Fasi

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

```
netapp/controller:24.10.1
netapp/restic:24.10.1
netapp/kopia:24.10.1
netapp/trident-autosupport:24.10.0
netapp/exechook:24.10.1
netapp/resourcebackup:24.10.1
netapp/resourcerestore:24.10.1
netapp/resourcedelete:24.10.1
bitnami/kubectl:1.30.2
kubebuilder/kube-rbac-proxy:v0.16.0
```

Ad esempio:

```
docker pull netapp/controller:24.10.1
```

```
docker tag netapp/controller:24.10.1 <private-registry-
url>/controller:24.10.1
```

```
docker push <private-registry-url>/controller:24.10.1
```

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. Creare un file denominato `protectValues.yaml`. Verificare che contenga le seguenti impostazioni di protezione Trident:

```
---
image:
  registry: <private-registry-url>
imagePullSecrets:
  - name: regcred
controller:
  image:
    registry: <private-registry-url>
rbacProxy:
  image:
    registry: <private-registry-url>
crCleanup:
  imagePullSecrets:
    - name: regcred
webhooksCleanup:
  imagePullSecrets:
    - name: regcred
```

7. Installare i CRD Trident Protect:

```
helm install trident-protect-crds netapp-trident-protect/trident-
protect-crds --version 100.2410.1 --create-namespace --namespace
trident-protect
```

8. Utilizzare Helm per installare Trident Protect utilizzando uno dei seguenti comandi. Sostituire `<name_of_cluster>` con un nome cluster, che verrà assegnato al cluster e utilizzato per identificare i backup e gli snapshot del cluster:
   - Installare Trident Protect normalmente:

```
helm install trident-protect netapp-trident-protect/trident-
protect --set clusterName=<name_of_cluster> --version 100.2410.1
--create-namespace --namespace trident-protect -f
protectValues.yaml
```

   - Installare Trident Protect e disattivare i caricamenti giornalieri programmati del pacchetto di supporto Trident Protect AutoSupport:

```
helm install trident-protect netapp-trident-protect/trident-
protect --set autoSupport.enabled=false --set
clusterName=<name_of_cluster> --version 100.2410.1 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

Specificare i limiti delle risorse del contenitore Trident Protect

È possibile utilizzare un file di configurazione per specificare i limiti delle risorse per i contenitori Trident Protect dopo l'installazione di Trident Protect. L'impostazione di limiti delle risorse consente di controllare la quantità di risorse del cluster utilizzata dalle operazioni Trident Protect.

Fasi

1. Creare un file denominato `resourceLimits.yaml`.
2. Popolare il file con opzioni di limite delle risorse per i contenitori Trident Protect in base alle esigenze dell'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 -f <resourceLimits.yaml>
--reuse-values
```

Installare il plugin Trident Protect CLI

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

Installare 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/24.10.1/tridentctl-protect-linux-amd64
```

Scarica il plugin per CPU Linux ARM64

Fasi

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-linux-arm64
```

Scarica il plugin per le CPU Mac AMD64

Fasi

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-macos-amd64
```

Scarica il plugin per le CPU Mac ARM64

Fasi

1. Scarica il plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-macos-arm64
```

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

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

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

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

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

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

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 alcuni comandi.

Attivare il completamento automatico per la shell Bash

Fasi

1. Scaricare lo script di completamento:

```
curl -L -O https://github.com/NetApp/tridentctl-
protect/releases/download/24.10.1/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. Scaricare lo script di completamento:

```
curl -L -O https://github.com/NetApp/tridentctl-
protect/releases/download/24.10.1/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.

Gestire Trident Protect

Gestire le autorizzazioni e il controllo degli accessi Trident Protect

Trident Protect utilizza il modello Kubernetes di role-based access control (RBAC). Per impostazione predefinita, Trident Protect fornisce un unico spazio dei nomi di sistema e l'account del servizio predefinito associato. Se hai un'organizzazione con molti utenti o esigenze di sicurezza specifiche, puoi utilizzare le funzionalità RBAC di Trident Protect per ottenere un controllo più granulare sull'accesso 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 devono avere accesso a oggetti risorse personalizzati protetti da Trident con privilegi, tra cui:
>
> - **AppVault**: Richiede i dati delle credenziali del bucket
> - **AutoSupportBundle**: Raccoglie metriche, registri e altri dati sensibili di 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

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

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.

Generare un bundle di supporto Trident Protect

Trident Protect consente agli amministratori di generare bundle che includono informazioni utili al supporto di NetApp, tra cui log, metriche e informazioni sulla topologia dei cluster e delle applicazioni gestiti. Se si è connessi a Internet, è possibile caricare pacchetti di supporto nel 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 `astra-support-bundle.yaml` file con i valori corretti, applicare la CR:

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

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>
```

Aggiornare Trident Protect

È possibile aggiornare Trident Protect alla versione più recente per sfruttare le nuove funzionalità o le correzioni dei bug.

Per aggiornare Trident Protect, procedere come segue.

Fasi

1. Aggiornare il repository di Trident Helm:

```
helm repo update
```

2. Aggiornare i CRD Trident Protect:

```
helm upgrade trident-protect-crds netapp-trident-protect/trident-
protect-crds --version 100.2410.1  --namespace trident-protect
```

3. Aggiornamento di Trident Protect:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2410.1 --namespace trident-protect
```

Gestisci e proteggi le applicazioni

Utilizzare gli oggetti Trident Protect AppVault per gestire i bucket

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

Esempi di generazione delle chiavi e di definizione di AppVault

Quando si definisce un CR AppVault, è necessario includere le credenziali per accedere alle risorse ospitate dal provider. La modalità di generazione delle chiavi per le credenziali varia a seconda del provider. Di seguito sono riportati alcuni esempi di generazione delle chiavi della riga di comando per diversi provider, seguiti da

esempi di definizioni AppVault per ciascun provider.

Esempi di generazione delle chiavi

Puoi utilizzare i seguenti esempi per creare chiavi per le credenziali di ciascun cloud provider.

Google Cloud

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

Amazon S3 (AWS)

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

Esempi di AppVault CR

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

Google Cloud

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  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-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

Microsoft Azure

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  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-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

ONTAP S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: OntapS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

StorageGRID S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket-b643cc50-0429-4ad5-
971f-ac4a83621922
  namespace: trident-protect
spec:
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

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.

Google Cloud

```
tridentctl-protect create vault GCP my-new-vault --bucket mybucket
--project my-gcp-project --secret <gcp-creds>/<credentials>
```

Amazon S3 (AWS)

```
tridentctl-protect create vault AWS <vault-name> --bucket <bucket-name>
--secret  <secret-name>  --endpoint <s3-endpoint>
```

Microsoft Azure

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

Generico S3

```
tridentctl-protect create vault GenericS3 <vault-name> --bucket
<bucket-name> --secret  <secret-name>  --endpoint <s3-endpoint>
```

ONTAP S3

```
tridentctl-protect create vault OntapS3 <vault-name> --bucket <bucket-
name> --secret  <secret-name>  --endpoint <s3-endpoint>
```

StorageGRID S3

```
tridentctl-protect create vault StorageGridS3 s3vault --bucket <bucket-
name> --secret  <secret-name>  --endpoint <s3-endpoint>
```

Utilizzare il browser AppVault per 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
   ```

   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 cluster nell'installazione di Trident Protect helm. Ad 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 tutte le istantanee e i backup memorizzati nel bucket associato.

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
```

Definire un'applicazione da gestire con Trident Protect

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

Creare un AppVault CR

È necessario creare una CR AppVault che verrà utilizzata quando si eseguono operazioni di protezione dei dati sull'applicazione e la CR AppVault deve risiedere nel cluster in cui è installato Trident Protect. AppVault CR è specifico per l'ambiente in uso; per esempi di CRS AppVault, fare riferimento a. "Risorse personalizzate AppVault."

Definire un'applicazione

È necessario definire ogni applicazione che si desidera gestire con Trident Protect. È possibile definire un'applicazione da gestire creando manualmente un CR di applicazione o utilizzando l'interfaccia 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 le etichette dello spazio dei nomi o un nome dello spazio dei nomi per specificare gli spazi dei nomi in cui esistono le risorse dell'applicazione. Lo spazio dei nomi dell'applicazione deve essere parte di questo elenco.

        Esempio YAML:

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

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

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

Aggiungere un'applicazione utilizzando l'interfaccia CLI

Fasi

1. Creare e applicare la definizione dell'applicazione, 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 nell'esempio seguente:

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-namespace>
```

Proteggi le applicazioni con Trident Protect

Puoi proteggere tutte le app gestite da Trident Protect creando snapshot e backup con policy di protezione automatizzate o su base ad-hoc.

> È possibile configurare Trident Protect per bloccare e sbloccare i file system durante le operazioni di protezione dei dati. "Ulteriori informazioni sulla configurazione del blocco del filesystem con Trident Protect".

Crea un'istantanea on-demand

Puoi creare uno snapshot on-demand in qualsiasi momento.

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.

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**: (*Optional*): Nome dello snapshot da utilizzare come origine del backup. Se non viene fornito, verrà creato e eseguito il backup di uno snapshot temporaneo.

```
---
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> -n
<application_namespace>
```

Creare un piano di data Protection

Una policy di protezione protegge un'applicazione creando snapshot, backup o entrambi in base a una pianificazione definita. È possibile scegliere di creare snapshot e backup ogni ora, ogni giorno, ogni settimana e ogni mese, nonché specificare il numero di copie da conservare.

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**: Il numero di backup da conservare. Zero indica che non è necessario creare backup.
   - **Spec.snapshotRetention**: Il numero di snapshot da conservare. Zero indica che non è necessario creare snapshot.
   - **spec.granularity**: frequenza di esecuzione della pianificazione. Valori possibili, insieme ai campi associati obbligatori:
     - `hourly` (è necessario specificare `spec.minute`)
     - `daily` (richiede di specificare `spec.minute` e `spec.hour`)
     - `weekly` (è necessario specificare `spec.minute, spec.hour`, e `spec.dayOfWeek`)
     - `monthly` (è necessario specificare `spec.minute, spec.hour`, e `spec.dayOfMonth`)
   - **Spec.dayOfMonth**: (*Optional*) il giorno del mese (1 - 31) in cui dovrebbe essere eseguito il programma. Questo campo è obbligatorio se la granularità è impostata su `monthly`.
   - **Spec.DayOfWeek**: (*Optional*) il giorno della settimana (0 - 7) in cui dovrebbe essere eseguito il programma. I valori di 0 o 7 indicano la domenica. Questo campo è obbligatorio se la granularità è impostata su `weekly`.
   - **Spec.hour**: (*Optional*) l'ora del giorno (0 - 23) in cui dovrebbe essere eseguito il programma. Questo campo è obbligatorio se la granularità è impostata su `daily`, , `weekly`o `monthly`.
   - **Spec.minute**: (*Optional*) il minuto dell'ora (0 - 59) che dovrebbe essere eseguito. Questo campo è obbligatorio se la granularità è impostata su `hourly`, , `daily` `weekly`o `monthly`.

```
---
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: <monthly>
  dayOfMonth: "1"
  dayOfWeek: "0"
  hour: "0"
  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> --data-mover
<kopia_or_restic> --day-of-month <day_of_month_to_run_schedule>
--day-of-week <day_of_month_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> -n <application_namespace>
```

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.

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 è stato installato Trident Protect, è possibile abilitare una funzionalità di backup e ripristino efficiente in termini di spazio per backend di storage che utilizzano la classe di storage Azure-NetApp-Files e che sono stati creati prima di Trident 24,06. Questa funzionalità funziona con volumi NFSv4 e non occupa spazio aggiuntivo dal pool di capacità.

Prima di iniziare

Verificare quanto segue:

- Trident Protect è stato installato.
- È stata definita un'applicazione in Trident Protect. Questa applicazione dispone di funzionalità di protezione limitate fino al completamento di 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 snapshot non è abilitata, Trident Protect sceglie la normale funzionalità di backup, che consuma temporaneamente spazio nel pool di capacità durante il processo di backup. In questo caso, verificare che nel pool di capacità sia disponibile spazio sufficiente per creare un volume temporaneo delle dimensioni del volume di cui si desidera eseguire il backup.

Risultato

L'applicazione è pronta per il backup e il ripristino utilizzando Trident Protect. Ciascun PVC è inoltre disponibile per essere utilizzato da altre applicazioni per backup e ripristini.

Ripristina le applicazioni utilizzando Trident Protect

Puoi utilizzare Trident Protect per ripristinare l'applicazione da uno snapshot o da un backup. Il ripristino da uno snapshot esistente sarà più rapido quando 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.

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 RedHat OpenShift, è importante notare il ruolo critico delle annotazioni dello spazio dei nomi negli ambienti OpenShift. Le annotazioni dello spazio dei nomi assicurano che i pod ripristinati aderiscano alle autorizzazioni e alle configurazioni di sicurezza appropriate definite dai vincoli del contesto di protezione OpenShift (SCC) e possano accedere ai volumi senza problemi di autorizzazione. Per ulteriori informazioni, fare riferimento alla "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:

```
kubectl set env -n trident-protect deploy/trident-protect-controller-
manager
RESTORE_SKIP_NAMESPACE_ANNOTATIONS=<annotation_key_to_skip_1>,<annotation_
key_to_skip_2>
```

Se l'applicazione di origine è stata installata utilizzando Helm con il `--create-namespace` flag, viene assegnato un trattamento speciale al `name` tasto etichetta. Durante il processo di ripristino o failover, Trident Protect copia questa etichetta nello spazio dei nomi di destinazione, ma aggiorna il valore allo spazio dei nomi di destinazione se il valore di origine corrisponde allo spazio dei nomi di origine. Se questo valore non corrisponde allo spazio dei nomi di origine, viene copiato nello spazio dei nomi di destinazione senza modifiche.

Esempio

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

Prima dell'operazione di ripristino o failover

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

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

Dopo l'operazione di ripristino

La tabella seguente illustra lo stato dello spazio dei nomi di destinazione di esempio dopo l'operazione di

ripristino o failover. Alcune chiavi sono state aggiunte, altre sono state sovrascritte e l' `name`etichetta è stata aggiornata per corrispondere allo spazio dei nomi di destinazione:

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

Ripristino da un backup a uno spazio dei nomi diverso

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

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.
   - **Spec.storageClassMapping**: Associazione della classe di archiviazione di origine dell'operazione di ripristino alla classe di archiviazione di destinazione. Sostituire `destinationStorageClass` e `sourceStorageClass` 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"}]
       storageClassMapping:
         destination: "${destinationStorageClass}"
         source: "${sourceStorageClass}"
     ```

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:
   - **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.

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:
   - **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.

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.

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> -n <application_namespace>
--context <destination_cluster_name>
```

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 uno spazio dei nomi diverso utilizzando una CR SnapshotRestore, Trident Protect ripristina l'applicazione in un nuovo spazio dei nomi e crea una CR dell'applicazione per l'applicazione ripristinata. Per proteggere l'applicazione ripristinata, creare backup o snapshot on-demand o stabilire una pianificazione della protezione.

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.
   - **Spec.storageClassMapping**: Associazione della classe di archiviazione di origine dell'operazione di ripristino alla classe di archiviazione di destinazione. Sostituire `destinationStorageClass` e `sourceStorageClass` 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"}]
       storageClassMapping:
         destination: "${destinationStorageClass}"
         source: "${sourceStorageClass}"
     ```

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:
   - **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.

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:
   - **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 <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}'
```

Replica le applicazioni utilizzando NetApp SnapMirror e Trident Protect

Utilizzando Trident Protect, puoi utilizzare le funzionalità di replica asincrona della tecnologia NetApp SnapMirror per replicare le modifiche ai dati e alle applicazioni da un backend storage a un 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 RedHat OpenShift, è importante notare il ruolo critico delle annotazioni dello spazio dei nomi negli ambienti OpenShift. Le annotazioni dello spazio dei nomi assicurano che i pod ripristinati aderiscano alle autorizzazioni e alle configurazioni di sicurezza appropriate definite dai vincoli del contesto di protezione OpenShift (SCC) e possano accedere ai volumi senza problemi di autorizzazione. Per ulteriori informazioni, fare riferimento alla "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:

```
kubectl set env -n trident-protect deploy/trident-protect-controller-
manager
RESTORE_SKIP_NAMESPACE_ANNOTATIONS=<annotation_key_to_skip_1>,<annotation_
key_to_skip_2>
```

Se l'applicazione di origine è stata installata utilizzando Helm con il `--create-namespace` flag, viene assegnato un trattamento speciale al `name` tasto etichetta. Durante il processo di ripristino o failover, Trident Protect copia questa etichetta nello spazio dei nomi di destinazione, ma aggiorna il valore allo spazio dei nomi di destinazione se il valore di origine corrisponde allo spazio dei nomi di origine. Se questo valore non corrisponde allo spazio dei nomi di origine, viene copiato nello spazio dei nomi di destinazione senza modifiche.

Esempio

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

Prima dell'operazione di ripristino o failover

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

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

Dopo l'operazione di ripristino

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

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

È possibile configurare Trident Protect per bloccare e sbloccare i file system durante le operazioni di protezione dei dati. "Ulteriori informazioni sulla configurazione del blocco del filesystem con Trident Protect".

Impostare una relazione di replica

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

- Scegliere la frequenza con cui desideri che Trident Protect crei un'istantanea dell'applicazione (che include le risorse Kubernetes dell'app e gli snapshot di 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. Creare un AppVault per l'applicazione di origine sul cluster 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>
```

2. 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. Se si desidera, creare un'istantanea dell'applicazione di origine. Questo snapshot viene utilizzato come base per l'applicazione nel cluster di destinazione. Se si salta questo passaggio, è necessario attendere l'esecuzione dello snapshot pianificato successivo in modo da disporre di uno snapshot recente.

Acquisire un'istantanea utilizzando una 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**: (*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 dell'applicazione di origine CR.
   - **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-0e1f88ab-f013-4bce-8ae9-6afed9df59a1
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  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
```

Scattare una snapshot utilizzando la CLI

a. 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>
```

4. Creare un'applicazione di origine AppVault CR sul cluster di destinazione che sia identica all'AppVault CR applicato sul cluster di origine e denominarla (ad esempio, `trident-protect-appvault-primary-destination.yaml`).
5. Applicare la CR:

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

6. Creare un AppVault 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 my-app-namespace
```

7. Creare un file CR AppMirrorRelationship:

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.storageClassName**: (*required*) scegliere il nome di una classe di archiviazione valida nel cluster. La classe di storage deve essere collegata a una macchina virtuale di storage ONTAP sottoposta a peering con l'ambiente di origine.
- **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. Creare e applicare l'oggetto AppMirrorRelationship, sostituendo i valori tra parentesi con le informazioni dell'ambiente. Ad esempio:

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirorrelationship> --destination-app-vault
<my_vault_name> --recurrence-rule <rule> --source-app
<my_source_app> --source-app-vault <my_source_app_vault> -n
<application_namespace>
```

8. (*Optional*) controllare 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

Con Trident Protect puoi eseguire il failover di applicazioni replicate su un cluster di destinazione. Questa procedura interrompe la relazione di replica e porta l'applicazione online sul cluster di destinazione. Trident Protect non interrompe l'applicazione sul cluster di origine se era operativa.

Fasi

1. Aprire il file AppMirrorRelationship CR (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. Creare un'istantanea dell'applicazione di origine.
2. Aprire il file AppMirrorRelationship CR (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. Eliminare la CR AppMirrorRelationship nel cluster di destinazione originale. 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 AppVault CRS sia pronto su ogni cluster.
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 di replica, Trident Protect sposta l'applicazione nel backend dello storage di destinazione, continuando nel contempo la replica nel back-end dello storage di origine. Trident Protect interrompe l'applicazione di origine e replica i dati sulla destinazione prima di eseguire il failover sull'app di destinazione.

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

Fasi

1. Creare un’istantanea 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. Al termine dell'istantanea, ottenere lo stato dell'istantanea:

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

3. Trovare il valore di **shutdownsnapshot.status.appArchivePath** utilizzando il comando seguente 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 cluster di destinazione al 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 replica originale, Trident Protect replica (risincronizza) tutte le modifiche apportate all'applicazione di origine prima di invertire la direzione di replica.

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. Eliminare la CR AppMirrorRelationship:

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

Migrazione delle applicazioni con Trident Protect

Puoi migrare le tue applicazioni tra cluster o classi di storage ripristinando i dati snapshot o di backup in un cluster o una classe di storage differenti.

> 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 in un cluster diverso (eseguire un clone tra cluster), creare un backup nel cluster di origine, quindi ripristinare il backup in 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 classe di archiviazione diversa ripristinando uno snapshot nella classe di archiviazione di destinazione differente.

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 i 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, a seconda del 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 filesystem. "Ulteriori informazioni sulla configurazione del blocco del filesystem 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.

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 di shell eseguibili.
- La dimensione dello script è limitata a 96 KB.
- Trident Protect utilizza le impostazioni di esecuzione hook e qualsiasi criterio corrispondente 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 gancio di esecuzione utilizza un'espressione regolare per far corrispondere i contenitori nel cluster. Quando un gancio corrisponde a un container, il gancio esegue lo script associato su quel container. Le espressioni regolari per i filtri utilizzano la sintassi RE2 (espressione regolare), che non supporta la creazione di un filtro che esclude i contenitori dall'elenco di 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 gancio di esecuzione personalizzato per un'applicazione utilizzando Trident Protect. Per creare gli hook di esecuzione, è necessario disporre delle autorizzazioni Owner (Proprietario), Admin (Amministratore) o Member (membro).

Utilizzare un CR

Fasi

1. Creare il file di risorse personalizzate (CR) e assegnargli un nome `trident-protect-hook.yaml`.
2. Configura i seguenti attributi per soddisfare la tua configurazione del cluster e dell'ambiente Trident Protect:

   - **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>
```

Disinstallare Trident Protect

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

Per rimuovere Trident Protect, procedere come segue.

Fasi

1. Rimuovere i file Trident Protect CR:

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

2. Rimozione di Trident Protect:

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

3. Rimuovere lo spazio dei nomi Trident Protect:

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

Informazioni sul copyright

Informazioni sul marchio commerciale