Gestire e proteggere le applicazioni

Trident

NetApp
January 15, 2026

Sommario

Gestire e proteggere le applicazioni

Utilizzare gli oggetti Trident Protect AppVault per gestire i bucket

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

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

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

Configurare l'autenticazione e le password di AppVault

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

Password del repository del data mover

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

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

Crea una password segreta per il repository del data mover

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

- A seconda del data mover utilizzato, è sufficiente includere la password corrispondente per quel data mover. Ad esempio, se utilizzi Restic e non prevedi di utilizzare Kopia in futuro, puoi includere solo la password Restic quando crei il segreto.
- Conservare la password in un luogo sicuro. Sarà necessaria per ripristinare i dati sullo stesso cluster o su uno diverso. Se il cluster o il `trident-protect` Se lo spazio dei nomi viene eliminato, non sarà possibile ripristinare i backup o gli snapshot senza la password.

Utilizzare un CR

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

Utilizzare la CLI

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

Autorizzazioni IAM di archiviazione compatibili con S3

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

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

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

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

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

Requisiti per l'identità del pod EKS con Trident Protect

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

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

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

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

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

Esempi di generazione di chiavi AppVault per i provider cloud

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

Google Cloud

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

Amazon S3 (AWS)

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

Microsoft Azure

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

S3 generico

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

ONTAP S3

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

StorageGrid S3

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

Esempi di creazione di AppVault

Di seguito sono riportati esempi di definizioni di AppVault per ciascun provider.

Esempi di CR di AppVault

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

- Facoltativamente, puoi specificare un segreto Kubernetes che contenga password personalizzate per la crittografia dei repository Restic e Kopia. Fare riferimento aPassword del repository del data mover per maggiori informazioni.
- Per gli oggetti Amazon S3 (AWS) AppVault, è possibile specificare facoltativamente un sessionToken, utile se si utilizza l'accesso Single Sign-On (SSO) per l'autenticazione. Questo token viene creato quando si generano le chiavi per il provider inEsempi di generazione di chiavi AppVault per i provider cloud .
- Per gli oggetti S3 AppVault, è possibile specificare facoltativamente un URL proxy di uscita per il traffico S3 in uscita utilizzando `spec.providerConfig.S3.proxyURL` chiave.

Google Cloud

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

Amazon S3 (AWS)

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

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

Microsoft Azure

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

S3 generico

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

ONTAP S3

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

StorageGrid S3

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

Esempi di creazione di AppVault utilizzando la CLI Trident Protect

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

- Facoltativamente, puoi specificare un segreto Kubernetes che contenga password personalizzate per la crittografia dei repository Restic e Kopia. Fare riferimento aPassword del repository del data mover per maggiori informazioni.
- Per gli oggetti S3 AppVault, è possibile specificare facoltativamente un URL proxy di uscita per il traffico S3 in uscita utilizzando `--proxy-url <ip_address:port>` discussione.

Google Cloud

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

Amazon S3 (AWS)

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

Microsoft Azure

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

S3 generico

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

ONTAP S3

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

StorageGrid S3

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

Visualizza le informazioni di AppVault

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

Passi

1. Visualizza il contenuto di un oggetto AppVault:

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

Esempio di output:

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

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

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

Rimuovere un AppVault

È possibile rimuovere un oggetto AppVault in qualsiasi momento.

> Non rimuovere il `finalizers` digitare nella CR di AppVault prima di eliminare l'oggetto AppVault. In tal caso, potrebbero esserci dati residui nel bucket AppVault e risorse orfane nel cluster.

Prima di iniziare

Assicurati di aver eliminato tutti gli snapshot e i CR di backup utilizzati dall'AppVault che desideri eliminare.

Rimuovere un AppVault utilizzando la CLI di Kubernetes

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

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

Definisci un'applicazione per la gestione con Trident Protect

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

Crea una CR AppVault

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

Definire un'applicazione

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

Aggiungere un'applicazione utilizzando un CR

Passi

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

Se è necessario applicare questa annotazione dopo che un'applicazione è già stata creata, è possibile utilizzare il seguente comando:

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

+
Esempio YAML:

+

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

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

   - **resourceFilter.resourceSelectionCriteria**: (Obbligatorio per il filtraggio) Utilizzare `Include` O `Exclude` per includere o escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti parametri resourceMatchers per definire le risorse da includere o escludere:

     - **resourceFilter.resourceMatchers**: un array di oggetti resourceMatcher. Se si definiscono più elementi in questo array, essi corrispondono come un'operazione OR e i campi all'interno di ciascun elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.
       - **resourceMatchers[].group**: (*Facoltativo*) Gruppo della risorsa da filtrare.
       - **resourceMatchers[].kind**: (*Facoltativo*) Tipo di risorsa da filtrare.
       - **resourceMatchers[].version**: (*Facoltativo*) Versione della risorsa da filtrare.
       - **resourceMatchers[].names**: (*Facoltativo*) Nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.

- **resourceMatchers[].namespaces**: (*Facoltativo*) Spazi dei nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].labelSelectors**: (*Facoltativo*) Stringa del selettore di etichetta nel campo metadata.name di Kubernetes della risorsa come definito in "Documentazione di Kubernetes" . Per esempio: `"trident.netapp.io/os=linux"` .

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

Per esempio:

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

2. Dopo aver creato la CR dell'applicazione adatta al tuo ambiente, applica la CR. Per esempio:

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

Passi

1. Crea e applica la definizione dell'applicazione utilizzando uno degli esempi seguenti, sostituendo i valori tra parentesi con le informazioni provenienti dal tuo ambiente. È possibile includere namespace e risorse nella definizione dell'applicazione utilizzando elenchi separati da virgole con gli argomenti mostrati negli esempi.

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

l'annotazione viene ignorata. Se non si utilizza l'annotazione, l'applicazione segue la"impostazione di congelamento globale Trident Protect" .

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

- Creare l'applicazione e utilizzare l'impostazione globale per il comportamento di blocco del file system:

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

- Creare l'applicazione e configurare le impostazioni dell'applicazione locale per il comportamento di blocco del file system:

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

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

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

Proteggi le applicazioni utilizzando Trident Protect

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

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

Crea uno snapshot on-demand

È possibile creare uno snapshot on-demand in qualsiasi momento.

Le risorse con ambito cluster vengono incluse in un backup, in uno snapshot o in un clone se sono esplicitamente referenziate nella definizione dell'applicazione o se contengono riferimenti a uno qualsiasi degli spazi dei nomi dell'applicazione.

Crea uno snapshot utilizzando un CR

Passi

1. Crea il file di risorse personalizzate (CR) e assegnagli un nome `trident-protect-snapshot-cr.yaml` .
2. Nel file creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.applicationRef**: Nome Kubernetes dell'applicazione di cui eseguire lo snapshot.
   - **spec.appVaultRef**: (*Obbligatorio*) Nome dell'AppVault in cui devono essere archiviati i contenuti dello snapshot (metadati).
   - **spec.reclaimPolicy**: (*Facoltativo*) Definisce cosa accade all'AppArchive di uno snapshot quando il CR dello snapshot viene eliminato. Ciò significa che anche quando impostato su `Retain` , lo snapshot verrà eliminato. Opzioni valide:
     - `Retain`(predefinito)
     - `Delete`

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: Snapshot
     metadata:
       namespace: my-app-namespace
       name: my-cr-name
     spec:
       applicationRef: my-application
       appVaultRef: appvault-name
       reclaimPolicy: Delete
     ```
3. Dopo aver popolato il `trident-protect-snapshot-cr.yaml` file con i valori corretti, applicare CR:

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

Creare uno snapshot utilizzando la CLI

Passi

1. Crea lo snapshot sostituendo i valori tra parentesi con le informazioni provenienti dal tuo ambiente. Per esempio:

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

Crea un backup su richiesta

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

> Le risorse con ambito cluster vengono incluse in un backup, in uno snapshot o in un clone se sono esplicitamente referenziate nella definizione dell'applicazione o se contengono riferimenti a uno qualsiasi degli spazi dei nomi dell'applicazione.

Prima di iniziare

Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di backup S3 di lunga durata. Se il token scade durante l'operazione di backup, l'operazione potrebbe non riuscire.

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

Crea un backup utilizzando un CR

Passi

1. Crea il file di risorse personalizzate (CR) e assegnagli un nome `trident-protect-backup-cr.yaml` .
2. Nel file creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.applicationRef**: (*Obbligatorio*) Nome Kubernetes dell'applicazione di cui eseguire il backup.
   - **spec.appVaultRef**: (*Obbligatorio*) Nome dell'AppVault in cui devono essere archiviati i contenuti del backup.
   - **spec.dataMover**: (*Facoltativo*) Una stringa che indica quale strumento di backup utilizzare per l'operazione di backup. Valori possibili (con distinzione tra maiuscole e minuscole):
     - `Restic`
     - `Kopia`(predefinito)
   - **spec.reclaimPolicy**: (*Facoltativo*) Definisce cosa accade a un backup quando viene rilasciato dalla sua richiesta. Valori possibili:
     - `Delete`
     - `Retain`(predefinito)
   - **spec.snapshotRef**: (*Facoltativo*): Nome dello snapshot da utilizzare come origine del backup. Se non specificato, verrà creato uno snapshot temporaneo e ne verrà eseguito il backup.
   - **metadata.annotations.protect.trident.netapp.io/full-backup** : (*Facoltativo*) Questa annotazione viene utilizzata per specificare se un backup deve essere non incrementale. Per impostazione predefinita, tutti i backup sono incrementali. Tuttavia, se questa annotazione è impostata su `true` , il backup diventa non incrementale. Se non specificato, il backup segue l'impostazione di backup incrementale predefinita. È consigliabile eseguire periodicamente un backup completo e poi eseguire backup incrementali tra un backup completo e l'altro, per ridurre al minimo i rischi associati ai ripristini.

     Esempio YAML:

```yaml
---
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
  annotations:
    protect.trident.netapp.io/full-backup: "true"
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 CR:

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

Creare un backup utilizzando la CLI

Passi

1. Crea il backup sostituendo i valori tra parentesi con le informazioni del tuo ambiente. Per esempio:

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

Facoltativamente puoi usare il `--full-backup` flag per specificare se un backup deve essere non incrementale. Per impostazione predefinita, tutti i backup sono incrementali. Quando si utilizza questo flag, il backup diventa non incrementale. È consigliabile eseguire periodicamente un backup completo e poi eseguire backup incrementali tra un backup completo e l'altro, per ridurre al minimo i rischi associati ai ripristini.

Creare un programma di protezione dei dati

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

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

Creare una pianificazione utilizzando un CR

Passi

1. Crea il file di risorse personalizzate (CR) e assegnagli un nome `trident-protect-schedule-cr.yaml` .
2. Nel file creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.dataMover**: (*Facoltativo*) Una stringa che indica quale strumento di backup utilizzare per l'operazione di backup. Valori possibili (con distinzione tra maiuscole e minuscole):
     - `Restic`
     - `Kopia`(predefinito)
   - **spec.applicationRef**: Nome Kubernetes dell'applicazione di cui eseguire il backup.
   - **spec.appVaultRef**: (*Obbligatorio*) Nome dell'AppVault in cui devono essere archiviati i contenuti del backup.
   - **spec.backupRetention**: Numero di backup da conservare. Zero indica che non devono essere creati backup (solo snapshot).
   - **spec.snapshotRetention**: Numero di snapshot da conservare. Zero indica che non devono essere creati snapshot.
   - **spec.granularity**: la frequenza con cui la pianificazione deve essere eseguita. Valori possibili, insieme ai campi associati obbligatori:
     - `Hourly`(richiede che tu specifichi `spec.minute`)
     - `Daily`(richiede che tu specifichi `spec.minute` E `spec.hour`)
     - `Weekly`(richiede che tu specifichi `spec.minute, spec.hour` , E `spec.dayOfWeek`)
     - `Monthly`(richiede che tu specifichi `spec.minute, spec.hour` , E `spec.dayOfMonth`)
     - `Custom`
   - **spec.dayOfMonth**: (*Facoltativo*) Il giorno del mese (1 - 31) in cui la pianificazione deve essere eseguita. Questo campo è obbligatorio se la granularità è impostata su `Monthly` . Il valore deve essere fornito come stringa.
   - **spec.dayOfWeek**: (*Facoltativo*) Il giorno della settimana (0 - 7) in cui deve essere eseguita la pianificazione. I valori 0 o 7 indicano domenica. Questo campo è obbligatorio se la granularità è impostata su `Weekly` . Il valore deve essere fornito come stringa.
   - **spec.hour**: (*Facoltativo*) L'ora del giorno (0 - 23) in cui la pianificazione deve essere eseguita. Questo campo è obbligatorio se la granularità è impostata su `Daily` , `Weekly` , O `Monthly` . Il valore deve essere fornito come stringa.
   - **spec.minute**: (*Facoltativo*) Il minuto dell'ora (0 - 59) in cui la pianificazione deve essere eseguita. Questo campo è obbligatorio se la granularità è impostata su `Hourly` , `Daily` , `Weekly` , O `Monthly` . Il valore deve essere fornito come stringa.
   - **metadata.annotations.protect.trident.netapp.io/full-backup-rule**: (*Facoltativo*) Questa annotazione viene utilizzata per specificare la regola per la pianificazione del backup completo. Puoi impostarlo su `always` per un backup completo costante o personalizzarlo in base alle tue esigenze. Ad esempio, se si sceglie la granularità giornaliera, è possibile specificare i giorni della

settimana in cui deve essere eseguito il backup completo.

Esempio di YAML per la pianificazione di backup e snapshot:

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

Esempio di YAML per la pianificazione solo snapshot:

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

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

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

Creare una pianificazione utilizzando la CLI

Passi

1. Crea la pianificazione della protezione, sostituendo i valori tra parentesi con le informazioni provenienti dal tuo ambiente. Per esempio:

   Puoi usare `tridentctl-protect create schedule --help` per visualizzare informazioni di aiuto dettagliate per questo comando.

   ```
   tridentctl-protect create schedule <my_schedule_name> --appvault
   <my_appvault_name> --app <name_of_app_to_snapshot> --backup
   -retention <how_many_backups_to_retain> --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>
   --full-backup-rule <string>
   ```

   Puoi impostare il `--full-backup-rule` bandiera a `always` per un backup completo costante o personalizzarlo in base alle tue esigenze. Ad esempio, se si sceglie la granularità giornaliera, è possibile specificare i giorni della settimana in cui deve essere eseguito il backup completo. Ad esempio, utilizzare `--full-backup-rule "Monday,Thursday"` per programmare un backup completo il lunedì e il giovedì.

   Per pianificazioni solo snapshot, impostare `--backup-retention 0` e specificare un valore maggiore di 0 per `--snapshot-retention` .

Elimina uno snapshot

Elimina gli snapshot pianificati o su richiesta di cui non hai più bisogno.

Passi

1. Rimuovere lo snapshot CR associato allo snapshot:

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

Elimina un backup

Elimina i backup pianificati o su richiesta di cui non hai più bisogno.

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

Passi

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.

Passi

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

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

Abilita backup e ripristino per le operazioni azure-netapp-files (ANF)

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

Prima di iniziare

Assicurarsi che:

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

Espandi per i passaggi di configurazione

1. Eseguire le seguenti operazioni in Trident se il volume ANF è stato creato prima dell'aggiornamento a Trident 24.10:

   a. Abilitare la directory snapshot per ogni PV basato su azure-netapp-files e associato all'applicazione:

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

   b. Verificare che la directory snapshot sia stata abilitata per ogni PV associato:

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

   Risposta:

   ```
   snapshotDirectory: "true"
   ```

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

Risultato

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

Ripristinare le applicazioni

Ripristina le applicazioni utilizzando Trident Protect

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

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

Ripristina da un backup a uno spazio dei nomi diverso

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

Il ripristino di un backup in uno spazio dei nomi diverso con risorse esistenti non modificherà le risorse che condividono i nomi con quelle presenti nel backup. Per ripristinare tutte le risorse nel backup, eliminare e ricreare lo spazio dei nomi di destinazione oppure ripristinare il backup in un nuovo spazio dei nomi.

Prima di iniziare

Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino S3 di lunga durata. Se il token scade durante l'operazione di ripristino, l'operazione potrebbe non riuscire.

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

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

Utilizzare un CR

Passi

1. Crea il file di risorse personalizzate (CR) e assegnagli un nome `trident-protect-backup-restore-cr.yaml` .
2. Nel file creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.appArchivePath**: il percorso all'interno di AppVault in cui sono archiviati i contenuti del backup. Per trovare questo percorso puoi usare il seguente comando:

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

   - **spec.appVaultRef**: (*Obbligatorio*) Nome dell'AppVault in cui sono archiviati i contenuti del backup.
   - **spec.namespaceMapping**: la 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 informazioni provenienti dal tuo ambiente.

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

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

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

   - **resourceFilter.resourceSelectionCriteria**: (Obbligatorio per il filtraggio) Utilizzare `Include` O `Exclude` per includere o escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti parametri resourceMatchers per definire le risorse da includere o escludere:
     - **resourceFilter.resourceMatchers**: un array di oggetti resourceMatcher. Se si definiscono più elementi in questo array, essi corrispondono come un'operazione OR e i campi all'interno di

ciascun elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.

- **resourceMatchers[].group**: (*Facoltativo*) Gruppo della risorsa da filtrare.
- **resourceMatchers[].kind**: (*Facoltativo*) Tipo di risorsa da filtrare.
- **resourceMatchers[].version**: (*Facoltativo*) Versione della risorsa da filtrare.
- **resourceMatchers[].names**: (*Facoltativo*) Nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].namespaces**: (*Facoltativo*) Spazi dei nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].labelSelectors**: (*Facoltativo*) Stringa del selettore di etichetta nel campo metadata.name di Kubernetes della risorsa come definito in "Documentazione di Kubernetes" . Per esempio: `"trident.netapp.io/os=linux"` .

  Per 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 CR:

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

Utilizzare la CLI

Passi

1. Ripristina il backup in uno spazio dei nomi diverso, sostituendo i valori tra parentesi con le informazioni del tuo ambiente. IL `namespace-mapping` l'argomento utilizza namespace separati da due punti per mappare i namespace di origine ai namespace di destinazione corretti nel formato `source1:dest1,source2:dest2` . Per esempio:

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

Ripristina da un backup allo spazio dei nomi originale

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

Prima di iniziare

Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino S3 di lunga durata. Se il token scade durante l'operazione di ripristino, l'operazione potrebbe non riuscire.

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

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

Utilizzare un CR

Passi

1. Crea il file di risorse personalizzate (CR) e assegnagli un nome `trident-protect-backup-ipr-cr.yaml` .
2. Nel file creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.appArchivePath**: il percorso all'interno di AppVault in cui sono archiviati i contenuti del backup. Per trovare questo percorso puoi usare il seguente comando:

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

   - **spec.appVaultRef**: (*Obbligatorio*) Nome dell'AppVault in cui sono archiviati i contenuti del backup.

     Per esempio:

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

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

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

   - **resourceFilter.resourceSelectionCriteria**: (Obbligatorio per il filtraggio) Utilizzare `Include` O `Exclude` per includere o escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti parametri resourceMatchers per definire le risorse da includere o escludere:
     - **resourceFilter.resourceMatchers**: un array di oggetti resourceMatcher. Se si definiscono più elementi in questo array, essi corrispondono come un'operazione OR e i campi all'interno di ciascun elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.
       - **resourceMatchers[].group**: (*Facoltativo*) Gruppo della risorsa da filtrare.
       - **resourceMatchers[].kind**: (*Facoltativo*) Tipo di risorsa da filtrare.

- **resourceMatchers[].version**: (*Facoltativo*) Versione della risorsa da filtrare.
- **resourceMatchers[].names**: (*Facoltativo*) Nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].namespaces**: (*Facoltativo*) Spazi dei nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].labelSelectors**: (*Facoltativo*) Stringa del selettore di etichetta nel campo metadata.name di Kubernetes della risorsa come definito in "Documentazione di Kubernetes" . Per esempio: `"trident.netapp.io/os=linux"` .

  Per 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 CR:

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

Utilizzare la CLI

Passi

1. Ripristina il backup nello spazio dei nomi originale, sostituendo i valori tra parentesi con le informazioni del tuo ambiente. IL `backup` l'argomento utilizza uno spazio dei nomi e un nome di backup nel formato `<namespace>/<name>` . Per esempio:

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

Ripristina da un backup a un cluster diverso

È possibile ripristinare un backup su un cluster diverso se si verifica un problema con il cluster originale.

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

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

Passi

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>
```

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

2. Visualizza il contenuto del 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 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 archivio:

Utilizzare un CR

1. Crea il file di risorse personalizzate (CR) e assegnagli un nome `trident-protect-backup-restore-cr.yaml` .
2. Nel file creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.appVaultRef**: (*Obbligatorio*) Nome dell'AppVault in cui sono archiviati i contenuti del backup.
   - **spec.appArchivePath**: il percorso all'interno di AppVault in cui sono archiviati i contenuti del backup. Per trovare questo percorso puoi usare 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 nel passaggio 2 per visualizzare il contenuto del backup.

   - **spec.namespaceMapping**: la 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 informazioni provenienti dal tuo ambiente.

     Per 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 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 provenienti dal proprio ambiente. L'argomento namespace-mapping utilizza namespace

separati da due punti per mappare i namespace di origine ai namespace di destinazione corretti nel formato source1:dest1,source2:dest2. Per esempio:

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

Ripristina da uno snapshot a uno spazio dei nomi diverso

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

> SnapshotRestore supporta il `spec.storageClassMapping` attributo, ma solo quando le classi di archiviazione di origine e di destinazione utilizzano lo stesso backend di archiviazione. Se si tenta di ripristinare un `StorageClass` che utilizza un backend di archiviazione diverso, l'operazione di ripristino non riuscirà.

Prima di iniziare

Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino S3 di lunga durata. Se il token scade durante l'operazione di ripristino, l'operazione potrebbe non riuscire.

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

Utilizzare un CR

Passi

1. Crea il file di risorse personalizzate (CR) e assegnagli un nome `trident-protect-snapshot-restore-cr.yaml` .
2. Nel file creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.appVaultRef**: (*Obbligatorio*) Nome dell'AppVault in cui sono archiviati i contenuti dello snapshot.
   - **spec.appArchivePath**: il percorso all'interno di AppVault in cui sono archiviati i contenuti dello snapshot. Per trovare questo percorso puoi usare il seguente comando:

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

   - **spec.namespaceMapping**: la 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 informazioni provenienti dal tuo ambiente.

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

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

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

   - **resourceFilter.resourceSelectionCriteria**: (Obbligatorio per il filtraggio) Utilizzare `Include` O `Exclude` per includere o escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti parametri resourceMatchers per definire le risorse da includere o escludere:
     - **resourceFilter.resourceMatchers**: un array di oggetti resourceMatcher. Se si definiscono più elementi in questo array, essi corrispondono come un'operazione OR e i campi all'interno di

ciascun elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.

- **resourceMatchers[].group**: (*Facoltativo*) Gruppo della risorsa da filtrare.
- **resourceMatchers[].kind**: (*Facoltativo*) Tipo di risorsa da filtrare.
- **resourceMatchers[].version**: (*Facoltativo*) Versione della risorsa da filtrare.
- **resourceMatchers[].names**: (*Facoltativo*) Nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].namespaces**: (*Facoltativo*) Spazi dei nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].labelSelectors**: (*Facoltativo*) Stringa del selettore di etichetta nel campo metadata.name di Kubernetes della risorsa come definito in "Documentazione di Kubernetes" . Per esempio: `"trident.netapp.io/os=linux"` .

  Per 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 CR:

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

Utilizzare la CLI

Passi

1. Ripristina lo snapshot in uno spazio dei nomi diverso, sostituendo i valori tra parentesi con le informazioni del tuo ambiente.

   - IL `snapshot` l'argomento utilizza uno spazio dei nomi e un nome di snapshot nel formato `<namespace>/<name>` .
   - IL `namespace-mapping` l'argomento utilizza namespace separati da due punti per mappare i

namespace di origine ai namespace di destinazione corretti nel formato `source1:dest1,source2:dest2` .

Per esempio:

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

Ripristina da uno snapshot allo spazio dei nomi originale

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

Prima di iniziare

Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino S3 di lunga durata. Se il token scade durante l'operazione di ripristino, l'operazione potrebbe non riuscire.

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

Utilizzare un CR

Passi

1. Crea il file di risorse personalizzate (CR) e assegnagli un nome `trident-protect-snapshot-ipr-cr.yaml` .
2. Nel file creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.appVaultRef**: (*Obbligatorio*) Nome dell'AppVault in cui sono archiviati i contenuti dello snapshot.
   - **spec.appArchivePath**: il percorso all'interno di AppVault in cui sono archiviati i contenuti dello snapshot. Per trovare questo percorso puoi usare 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. (*Facoltativo*) Se è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda le risorse contrassegnate con etichette particolari:

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

   - **resourceFilter.resourceSelectionCriteria**: (Obbligatorio per il filtraggio) Utilizzare `Include` O `Exclude` per includere o escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti parametri resourceMatchers per definire le risorse da includere o escludere:
     - **resourceFilter.resourceMatchers**: un array di oggetti resourceMatcher. Se si definiscono più elementi in questo array, essi corrispondono come un'operazione OR e i campi all'interno di ciascun elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.
       - **resourceMatchers[].group**: (*Facoltativo*) Gruppo della risorsa da filtrare.
       - **resourceMatchers[].kind**: (*Facoltativo*) Tipo di risorsa da filtrare.
       - **resourceMatchers[].version**: (*Facoltativo*) Versione della risorsa da filtrare.

- **resourceMatchers[].names**: (*Facoltativo*) Nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].namespaces**: (*Facoltativo*) Spazi dei nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].labelSelectors**: (*Facoltativo*) Stringa del selettore di etichetta nel campo metadata.name di Kubernetes della risorsa come definito in "Documentazione di Kubernetes" . Per esempio: `"trident.netapp.io/os=linux"` .

  Per 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 CR:

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

Utilizzare la CLI

Passi

1. Ripristina lo snapshot nello spazio dei nomi originale, sostituendo i valori tra parentesi con le informazioni del tuo ambiente. Per 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.

Passi

1. Utilizzare il seguente comando per recuperare lo stato dell'operazione di ripristino, sostituendo i valori tra parentesi con le informazioni provenienti dal proprio ambiente:

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

Utilizza le impostazioni di ripristino avanzate Trident Protect

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

Annotazioni e etichette dello spazio dei nomi durante le operazioni di ripristino e failover

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

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

È possibile impedire che annotazioni specifiche nello spazio dei nomi di destinazione vengano sovrascritte impostando la variabile di ambiente Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` prima di eseguire l'operazione di ripristino o failover. Per esempio:

```
helm upgrade trident-protect --set
restoreSkipNamespaceAnnotations=<annotation_key_to_skip_1>,<annotation_key
_to_skip_2> --reuse-values
```

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

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

Esempio

L'esempio seguente presenta uno spazio dei nomi di origine e di 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:

Spazio dei nomi	Annotazioni	Etichette
Namespace ns-1 (fonte)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• ambiente=produzione • conformità=hipaa • nome=ns-1
Namespace ns-2 (destinazione)	• annotation.one/key: "true" • annotazione.tre/chiave: "falso"	• 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, alcune sono state sovrascritte e `name` l'etichetta è stata aggiornata per corrispondere allo spazio dei nomi di destinazione:

Spazio dei nomi	Annotazioni	Etichette
Namespace ns-2 (destinazione)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotazione.tre/chiave: "falso"	• nome=ns-2 • conformità=hipaa • ambiente=produzione • ruolo=database

Campi supportati

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

Mappatura delle classi di archiviazione

IL `spec.storageClassMapping` L'attributo definisce una mappatura da una classe di archiviazione presente nell'applicazione di origine a una nuova classe di archiviazione nel cluster di destinazione. È possibile

utilizzarlo durante la migrazione di applicazioni tra cluster con classi di archiviazione diverse o quando si modifica il backend di archiviazione per le operazioni BackupRestore.

Esempio:

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

Annotazioni supportate

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

Annotazione	Tipo	Descrizione	Valore predefinito
protect.trident.netapp.io/data-mover-timeout-sec	corda	Tempo massimo (in secondi) consentito per l'interruzione dell'operazione di spostamento dei dati.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	corda	Limite massimo di dimensione (in megabyte) per la cache dei contenuti di Kopia.	"1000"

Replicare le applicazioni utilizzando NetApp SnapMirror e Trident Protect

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

Annotazioni e etichette dello spazio dei nomi durante le operazioni di ripristino e failover

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

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

È possibile impedire che annotazioni specifiche nello spazio dei nomi di destinazione vengano sovrascritte impostando la variabile di ambiente Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` prima di eseguire l'operazione di ripristino o failover. Per esempio:

```
helm upgrade trident-protect --set
restoreSkipNamespaceAnnotations=<annotation_key_to_skip_1>,<annotation_key
_to_skip_2> --reuse-values
```

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

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

Esempio

L'esempio seguente presenta uno spazio dei nomi di origine e di 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:

Spazio dei nomi	Annotazioni	Etichette
Namespace ns-1 (fonte)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• ambiente=produzione • conformità=hipaa • nome=ns-1

Spazio dei nomi	Annotazioni	Etichette
Namespace ns-2 (destinazione)	• annotation.one/key: "true" • annotazione.tre/chiave: "falso"	• 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, alcune sono state sovrascritte e `name` l'etichetta è stata aggiornata per corrispondere allo spazio dei nomi di destinazione:

Spazio dei nomi	Annotazioni	Etichette
Namespace ns-2 (destinazione)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotazione.tre/chiave: "falso"	• nome=ns-2 • conformità=hipaa • ambiente=produzione • ruolo=database

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

Hook di esecuzione durante le operazioni di failover e reverse

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

- Durante il failover, gli hook di esecuzione vengono copiati automaticamente dal cluster di origine al cluster di destinazione. Non è necessario ricrearli manualmente. Dopo il failover, gli hook di esecuzione sono presenti nell'applicazione e verranno eseguiti durante tutte le azioni rilevanti.
- Durante la sincronizzazione inversa o la risincronizzazione inversa, tutti gli hook di esecuzione esistenti sull'applicazione vengono rimossi. Quando l'applicazione di origine diventa l'applicazione di destinazione, questi hook di esecuzione non sono validi e vengono eliminati per impedirne l'esecuzione.

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

Impostare una relazione di replicazione

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

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

Passi

1. Nel cluster di origine, creare un AppVault per l'applicazione di origine. A seconda del provider di archiviazione, modifica un esempio in"Risorse personalizzate di AppVault" per adattarsi al tuo ambiente:

Crea un AppVault utilizzando una CR

a. Crea il file di risorse personalizzate (CR) e assegnagli un nome (ad esempio, `trident-protect-appvault-primary-source.yaml`).

b. Configurare i seguenti attributi:

- **metadata.name**: (*Obbligatorio*) Nome della risorsa personalizzata di AppVault. Prendi nota del nome scelto, perché altri file CR necessari per una relazione di replica fanno riferimento a questo valore.
- **spec.providerConfig**: (*Obbligatorio*) 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. Prendi nota dei valori scelti, perché altri file CR necessari per una relazione di replicazione fanno riferimento a questi valori. Fare riferimento a"Risorse personalizzate di AppVault" per esempi di CR AppVault con altri provider.
- **spec.providerCredentials**: (*Obbligatorio*) Memorizza i riferimenti a qualsiasi credenziale richiesta per accedere ad AppVault utilizzando il provider specificato.
  - **spec.providerCredentials.valueFromSecret**: (*Obbligatorio*) Indica che il valore delle credenziali deve provenire da un segreto.
    - **key**: (*Obbligatorio*) La chiave valida del segreto da cui effettuare la selezione.
    - **name**: (*Obbligatorio*) Nome del segreto contenente il valore per questo campo. Devono trovarsi nello stesso namespace.
  - **spec.providerCredentials.secretAccessKey**: (*Obbligatorio*) La chiave di accesso utilizzata per accedere al provider. Il **nome** deve corrispondere a **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*Obbligatorio*) Determina chi fornisce il backup; ad esempio, NetApp ONTAP S3, S3 generico, Google Cloud o Microsoft Azure. Valori possibili:
  - aws
  - azzurro
  - gcp
  - generico-s3
  - ontap-s3
  - storagegrid-s3

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

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

Creare un AppVault utilizzando la CLI

a. Crea AppVault, sostituendo i valori tra parentesi con le informazioni del tuo ambiente:

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

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

Creare l'applicazione sorgente utilizzando un CR

a. Crea il file di risorse personalizzate (CR) e assegnagli un nome (ad esempio, `trident-protect-app-source.yaml`).

b. Configurare i seguenti attributi:

- **metadata.name**: (*Obbligatorio*) Nome della risorsa personalizzata dell'applicazione. Prendi nota del nome scelto, perché altri file CR necessari per una relazione di replica fanno riferimento a questo valore.
- **spec.includedNamespaces**: (*Obbligatorio*) Un array di namespace ed etichette associate. Utilizzare i nomi degli spazi dei nomi e, facoltativamente, restringere l'ambito degli spazi dei nomi con etichette per specificare le risorse presenti negli spazi dei nomi elencati qui. 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 CR:

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

Creare l'applicazione sorgente utilizzando la CLI

a. Creare l'applicazione sorgente. Per esempio:

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

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

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

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

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

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

Creare la pianificazione della replica utilizzando un CR

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

i. Crea il file di risorse personalizzate (CR) e assegnagli un nome (ad esempio, `trident-protect-schedule.yaml`).

ii. Configurare i seguenti attributi:

- **metadata.name**: (*Obbligatorio*) Nome della risorsa personalizzata della pianificazione.
- **spec.appVaultRef**: (*Obbligatorio*) Questo valore deve corrispondere al campo metadata.name dell'AppVault per l'applicazione di origine.
- **spec.applicationRef**: (*Obbligatorio*) Questo valore deve corrispondere al campo metadata.name del CR dell'applicazione di origine.
- **spec.backupRetention**: (*Obbligatorio*) 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**: definisce una data di inizio in ora UTC e un intervallo di ricorrenza.
- **spec.snapshotRetention**: Deve essere impostato su 2.

Esempio YAML:

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

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

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

Creare la pianificazione della replica utilizzando la CLI

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

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

Esempio:

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

5. Nel cluster di destinazione, crea un'applicazione di origine AppVault CR identica all'AppVault CR applicata nel cluster di origine e assegnale un nome (ad esempio, `trident-protect-appvault-primary-destination.yaml`).
6. Applicare il CR:

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

7. Creare un CR AppVault di destinazione per l'applicazione di destinazione sul cluster di destinazione. A seconda del provider di archiviazione, modificare un esempio in"Risorse personalizzate di AppVault" per adattarsi al tuo ambiente:
   a. Crea il file di risorse personalizzate (CR) e assegnagli un nome (ad esempio, `trident-protect-appvault-secondary-destination.yaml`).
   b. Configurare i seguenti attributi:
      - **metadata.name**: (*Obbligatorio*) Nome della risorsa personalizzata di AppVault. Prendi nota del nome scelto, perché altri file CR necessari per una relazione di replica fanno riferimento a questo valore.
      - **spec.providerConfig**: (*Obbligatorio*) Memorizza la configurazione necessaria per accedere ad AppVault utilizzando il provider specificato. Scegli un `bucketName` e qualsiasi altro dettaglio necessario per il tuo fornitore. Prendi nota dei valori scelti, perché altri file CR necessari per una relazione di replicazione fanno riferimento a questi valori. Fare riferimento a"Risorse personalizzate di AppVault" per esempi di CR AppVault con altri provider.
      - **spec.providerCredentials**: (*Obbligatorio*) Memorizza i riferimenti a qualsiasi credenziale richiesta per accedere ad AppVault utilizzando il provider specificato.
        - **spec.providerCredentials.valueFromSecret**: (*Obbligatorio*) Indica che il valore delle

credenziali deve provenire da un segreto.

- **key**: (*Obbligatorio*) La chiave valida del segreto da cui effettuare la selezione.
- **name**: (*Obbligatorio*) Nome del segreto contenente il valore per questo campo. Devono trovarsi nello stesso namespace.

- **spec.providerCredentials.secretAccessKey**: (*Obbligatorio*) La chiave di accesso utilizzata per accedere al provider. Il **nome** deve corrispondere a **spec.providerCredentials.valueFromSecret.name**.

- **spec.providerType**: (*Obbligatorio*) Determina chi fornisce il backup; ad esempio, NetApp ONTAP S3, S3 generico, Google Cloud o Microsoft Azure. Valori possibili:
  - aws
  - azzurro
  - gcp
  - generico-s3
  - ontap-s3
  - storagegrid-s3

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

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

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

Creare un AppMirrorRelationship utilizzando un CR

a. Crea il file di risorse personalizzate (CR) e assegnagli un nome (ad esempio, `trident-protect-relationship.yaml`).

b. Configurare i seguenti attributi:

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

  Esempio YAML:

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

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

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

Creare un AppMirrorRelationship utilizzando la CLI

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

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirorrelationship> --destination-app-vault
<my_vault_name> --source-app-vault <my_vault_name> --recurrence
-rule <rule> --namespace-mapping <ns_mapping> --source-app-id
<source_app_UID> --source-app <my_source_app_name> --storage
-class <storage_class_name> -n <application_namespace>
```

Esempio:

```
tridentctl-protect create appmirrorrelationship my-amr
--destination-app-vault appvault2 --source-app-vault appvault1
--recurrence-rule
"DTSTART:20220101T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5"
--source-app my-app --namespace-mapping "my-source-ns1:my-dest-
ns1,my-source-ns2:my-dest-ns2" --source-app-id 373f24c1-5769-
404c-93c3-5538af6ccc36 --storage-class my-storage-class -n my-
dest-ns1
```

9. (*Facoltativo*) Nel cluster di destinazione, controllare lo stato e la relazione di replica:

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

Failover sul cluster di destinazione

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

Passi

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

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

4. (*Facoltativo*) Creare tutte le pianificazioni di protezione necessarie sull'applicazione sottoposta a failover.
5. (*Facoltativo*) Controllare lo stato e la situazione della relazione di replicazione:

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

Risincronizzare una relazione di replicazione 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 ignorate.

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

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

Passi

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

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

5. Se sono state create delle pianificazioni di protezione sul cluster di destinazione per proteggere l'applicazione sottoposta a failover, rimuoverle. Tutte le pianificazioni rimanenti causano errori negli snapshot del volume.

Risincronizzazione inversa di una relazione di replicazione non riuscita

Quando si esegue la risincronizzazione inversa di una relazione di replicazione 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.

Passi

1. Nel cluster di destinazione originale, eliminare il CR AppMirrorRelationship. Ciò fa sì che la destinazione diventi la sorgente. Se sul nuovo cluster di destinazione sono presenti pianificazioni di protezione rimanenti, rimuoverle.
2. Imposta una relazione di replica applicando i file CR utilizzati originariamente per impostare la relazione ai cluster opposti.
3. Assicurarsi che la nuova destinazione (cluster di origine originale) sia configurata con entrambi i CR di AppVault.
4. Impostare una relazione di replicazione sul cluster opposto, configurando i valori per la direzione inversa.

Invertire la direzione di replicazione dell'applicazione

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

In questa situazione si scambiano la sorgente e la destinazione.

Passi

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

Creare uno snapshot di arresto utilizzando un CR

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

b. Creare un file CR ShutdownSnapshot:

   i. Crea il file di risorse personalizzate (CR) e assegnagli un nome (ad esempio, `trident-protect-shutdownsnapshot.yaml`).

   ii. Configurare i seguenti attributi:

   - **metadata.name**: (*Obbligatorio*) Nome della risorsa personalizzata.
   - **spec.AppVaultRef**: (*Obbligatorio*) Questo valore deve corrispondere al campo metadata.name dell'AppVault per l'applicazione di origine.
   - **spec.ApplicationRef**: (*Obbligatorio*) Questo valore deve corrispondere al campo metadata.name del 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 CR:

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

Creare uno snapshot di arresto utilizzando la CLI

a. Crea lo snapshot di arresto, sostituendo i valori tra parentesi con le informazioni provenienti dal tuo ambiente. Per esempio:

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

2. Nel cluster di origine, una volta completato lo snapshot di arresto, ottenere lo stato dello snapshot di arresto:

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

3. Nel cluster di origine, trova il valore di **shutdownsnapshot.status.appArchivePath** utilizzando il seguente comando e registra l'ultima parte del percorso del file (chiamato anche basename; sarà tutto ciò che segue l'ultima barra):

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

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

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

5. Eseguire i passaggi di risincronizzazione inversa inRisincronizzazione inversa di una relazione di replicazione non riuscita .
6. Abilitare le pianificazioni di protezione sul nuovo cluster di origine.

Risultato

A causa della replicazione inversa si verificano le seguenti azioni:

- Viene creato uno snapshot delle risorse Kubernetes dell'app sorgente originale.
- I pod dell'app sorgente originale vengono arrestati correttamente eliminando le risorse Kubernetes dell'app (lasciando in posizione PVC e PV).
- Dopo aver arrestato i pod, vengono acquisiti e replicati gli snapshot dei volumi dell'app.
- Le relazioni SnapMirror vengono interrotte, rendendo i volumi di destinazione pronti per la lettura/scrittura.
- Le risorse Kubernetes dell'app vengono ripristinate dallo snapshot precedente all'arresto, utilizzando i dati del volume replicati dopo l'arresto dell'app sorgente originale.
- La replicazione viene ristabilita nella direzione inversa.

Eseguire il failback delle applicazioni al cluster di origine originale

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

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

- Inizia con uno stato di failover.

- Risincronizza inversamente la relazione di replicazione.

> Non eseguire una normale operazione di risincronizzazione, poiché ciò eliminerà i dati scritti nel cluster di destinazione durante la procedura di failover.

- Invertire la direzione della replicazione.

Passi

1. Eseguire il Risincronizzazione inversa di una relazione di replicazione non riuscita passi.
2. Eseguire il Invertire la direzione di replicazione dell'applicazione passi.

Elimina una relazione di replicazione

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

Passi

1. Nel cluster di destinazione corrente, eliminare il CR AppMirrorRelationship:

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

Migrare le applicazioni utilizzando Trident Protect

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

> Quando si esegue la migrazione di un'applicazione, tutti gli hook di esecuzione configurati per l'applicazione vengono migrati insieme all'applicazione. Se è presente un hook di esecuzione post-ripristino, questo 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, creare uno snapshot o un backup e ripristinare i dati nello stesso cluster.

Passi

1. Eseguire una delle seguenti operazioni:
   a. "Crea uno snapshot".
   b. "Crea un backup".
2. Sullo stesso cluster, esegui una delle seguenti operazioni, a seconda che tu abbia creato uno snapshot o un backup:

a. "Ripristina i tuoi dati dallo snapshot".

b. "Ripristina i tuoi dati dal backup".

Clona in cluster diversi

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

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

Passi

1. "Crea un backup".
2. Assicurarsi che il CR di AppVault per il bucket di archiviazione degli oggetti che contiene il backup sia stato configurato sul cluster di destinazione.
3. Sul cluster di destinazione,"ripristina i tuoi dati dal backup" .

Migrare le applicazioni da una classe di archiviazione a un'altra classe di archiviazione

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

Ad esempio (escludendo i segreti dal ripristino CR):

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

Ripristinare lo snapshot utilizzando un CR

Passi

1. Crea il file di risorse personalizzate (CR) e assegnagli un nome `trident-protect-snapshot-restore-cr.yaml` .
2. Nel file creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.appArchivePath**: il percorso all'interno di AppVault in cui sono archiviati i contenuti dello snapshot. Per trovare questo percorso puoi usare il seguente comando:

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

   - **spec.appVaultRef**: (*Obbligatorio*) Nome dell'AppVault in cui sono archiviati i contenuti dello snapshot.
   - **spec.namespaceMapping**: la 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 informazioni provenienti dal tuo ambiente.

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

3. Facoltativamente, se è necessario selezionare solo determinate risorse dell'applicazione da ripristinare, aggiungere un filtro che includa o escluda le risorse contrassegnate con etichette particolari:
   - **resourceFilter.resourceSelectionCriteria**: (obbligatorio per il filtraggio) Usa `include or exclude` per includere o escludere una risorsa definita in resourceMatchers. Aggiungere i seguenti parametri resourceMatchers per definire le risorse da includere o escludere:
     - **resourceFilter.resourceMatchers**: un array di oggetti resourceMatcher. Se si definiscono più elementi in questo array, essi corrispondono come un'operazione OR e i campi all'interno di ciascun elemento (gruppo, tipo, versione) corrispondono come un'operazione AND.
       - **resourceMatchers[].group**: (*Facoltativo*) Gruppo della risorsa da filtrare.
       - **resourceMatchers[].kind**: (*Facoltativo*) Tipo di risorsa da filtrare.

- **resourceMatchers[].version**: (*Facoltativo*) Versione della risorsa da filtrare.
- **resourceMatchers[].names**: (*Facoltativo*) Nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].namespaces**: (*Facoltativo*) Spazi dei nomi nel campo metadata.name di Kubernetes della risorsa da filtrare.
- **resourceMatchers[].labelSelectors**: (*Facoltativo*) Stringa del selettore di etichetta nel campo metadata.name di Kubernetes della risorsa come definito in "Documentazione di Kubernetes" . Per esempio: `"trident.netapp.io/os=linux"` .

  Per 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 CR:

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

Ripristinare lo snapshot utilizzando la CLI

Passi

1. Ripristina lo snapshot in uno spazio dei nomi diverso, sostituendo i valori tra parentesi con le informazioni del tuo ambiente.

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

     Per esempio:

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

Gestire gli hook di esecuzione Trident Protect

Un hook di esecuzione è un'azione personalizzata che è possibile configurare per essere eseguita insieme a un'operazione di protezione dei dati di un'app gestita. Ad esempio, se si dispone di un'app di database, è possibile utilizzare un hook 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 ganci di esecuzione

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

- Pre-istantanea
- Post-istantanea
- Pre-backup
- Post-backup
- Post-ripristino
- Post-failover

Ordine di esecuzione

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

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

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

1. Eseguiti i pre-snapshot hook
2. Eseguiti i ganci 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.

Dovresti sempre testare gli script di esecuzione prima di abilitarli in un ambiente di produzione. È possibile utilizzare il comando 'kubectl exec' per testare comodamente gli script. Dopo aver abilitato gli hook di esecuzione in un ambiente di produzione, testare gli snapshot e i backup risultanti per assicurarsi che siano coerenti. È possibile farlo clonando l'app in uno spazio dei nomi temporaneo, ripristinando lo snapshot o il backup e quindi testando l'app.

Se un hook di esecuzione pre-snapshot aggiunge, modifica o rimuove risorse Kubernetes, tali modifiche vengono incluse nello snapshot o nel backup e in qualsiasi successiva operazione di ripristino.

Note importanti sui ganci di esecuzione personalizzati

Quando pianifichi gli hook di esecuzione per le tue app, tieni presente quanto segue.

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

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

Filtri di hook di esecuzione

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

Ogni filtro aggiunto a un hook di esecuzione utilizza un'espressione regolare per abbinare i contenitori nel

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

> Se si aggiunge un filtro namespace a un hook di esecuzione eseguito dopo un'operazione di ripristino o clonazione e l'origine e la destinazione del ripristino o della clonazione si trovano in namespace diversi, il filtro namespace viene applicato solo al namespace di destinazione.

Esempi di hook di esecuzione

Visita il "Progetto GitHub NetApp Verda" per scaricare veri e propri hook di esecuzione per app popolari come Apache Cassandra ed Elasticsearch. Puoi anche vedere esempi e trarre spunti per strutturare i tuoi hook di esecuzione personalizzati.

Creare un hook di esecuzione

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

Utilizzare un CR

Passi

1. Crea il file di risorse personalizzate (CR) e assegnagli un nome `trident-protect-hook.yaml` .
2. Configurare i seguenti attributi in modo che corrispondano all'ambiente Trident Protect e alla configurazione del cluster:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.applicationRef**: (*Obbligatorio*) Nome Kubernetes dell'applicazione per cui eseguire l'hook di esecuzione.
   - **spec.stage**: (*Obbligatorio*) Una stringa che indica in quale fase dell'azione deve essere eseguito l'hook di esecuzione. Valori possibili:
     - Pre
     - Inviare
   - **spec.action**: (*Obbligatorio*) Una stringa che indica quale azione intraprenderà l'hook di esecuzione, supponendo che tutti i filtri dell'hook di esecuzione specificati corrispondano. Valori possibili:
     - Istantanea
     - Backup
     - Ripristinare
     - Failover
   - **spec.enabled**: (*Facoltativo*) Indica se questo hook di esecuzione è abilitato o disabilitato. Se non specificato, il valore predefinito è true.
   - **spec.hookSource**: (*Obbligatorio*) Una stringa contenente lo script hook codificato in base64.
   - **spec.timeout**: (*Facoltativo*) Un numero che definisce per quanti minuti è consentita l'esecuzione dell'hook. Il valore minimo è 1 minuto e il valore predefinito è 25 minuti se non specificato.
   - **spec.arguments**: (*Facoltativo*) Un elenco YAML di argomenti che è possibile specificare per l'hook di esecuzione.
   - **spec.matchingCriteria**: (*Facoltativo*) Un elenco facoltativo di coppie chiave-valore di criteri, ciascuna delle quali costituisce un filtro di hook di esecuzione. È possibile aggiungere fino a 10 filtri per ogni hook di esecuzione.
   - **spec.matchingCriteria.type**: (*Facoltativo*) Una stringa che identifica il tipo di filtro dell'hook di esecuzione. Valori possibili:
     - Immagine contenitore
     - NomeContenitore
     - NomePod
     - Etichetta Pod
     - Nome dello spazio dei nomi
   - **spec.matchingCriteria.value**: (*Facoltativo*) Una stringa o espressione regolare che identifica il valore del filtro dell'hook di esecuzione.

     Esempio YAML:

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

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

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

Utilizzare la CLI

Passi

1. Crea l'hook di esecuzione, sostituendo i valori tra parentesi con le informazioni provenienti dal tuo ambiente. Per esempio:

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

Eseguire manualmente un hook di esecuzione

È possibile eseguire manualmente un hook di esecuzione per scopi di test o se è necessario rieseguirlo manualmente dopo un errore. Per eseguire manualmente gli hook di esecuzione è necessario disporre delle autorizzazioni di Proprietario, Amministratore o Membro.

L'esecuzione manuale di un hook di esecuzione consiste in due passaggi fondamentali:

1. Crea un backup delle risorse, che raccoglie le risorse e ne crea un backup, determinando dove verrà eseguito l'hook
2. Eseguire l'hook di esecuzione sul backup

Passaggio 1: creare un backup delle risorse

Utilizzare un CR

Passi

1. Crea il file di risorse personalizzate (CR) e assegnagli un nome `trident-protect-resource-backup.yaml` .
2. Configurare i seguenti attributi in modo che corrispondano all'ambiente Trident Protect e alla configurazione del cluster:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.applicationRef**: (*Obbligatorio*) Nome Kubernetes dell'applicazione per cui creare il backup delle risorse.
   - **spec.appVaultRef**: (*Obbligatorio*) Nome dell'AppVault in cui sono archiviati i contenuti del backup.
   - **spec.appArchivePath**: il percorso all'interno di AppVault in cui sono archiviati i contenuti del backup. Per trovare questo percorso puoi usare il seguente comando:

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

     Esempio YAML:

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

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

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

Utilizzare la CLI

Passi

1. Crea il backup sostituendo i valori tra parentesi con le informazioni del tuo ambiente. Per esempio:

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

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

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

3. Verificare che il backup sia andato a buon fine:

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

Passaggio 2: eseguire l’hook di esecuzione

Utilizzare un CR

Passi

1. Crea il file di risorse personalizzate (CR) e assegnagli un nome `trident-protect-hook-run.yaml` .
2. Configurare i seguenti attributi in modo che corrispondano all'ambiente Trident Protect e alla configurazione del cluster:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.applicationRef**: (*Obbligatorio*) Assicurati che questo valore corrisponda al nome dell'applicazione dal CR ResourceBackup creato nel passaggio 1.
   - **spec.appVaultRef**: (*Obbligatorio*) Assicurati che questo valore corrisponda all'appVaultRef del CR ResourceBackup creato nel passaggio 1.
   - **spec.appArchivePath**: assicurati che questo valore corrisponda all'appArchivePath del CR ResourceBackup creato nel passaggio 1.

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

   - **spec.action**: (*Obbligatorio*) Una stringa che indica quale azione intraprenderà l'hook di esecuzione, supponendo che tutti i filtri dell'hook di esecuzione specificati corrispondano. Valori possibili:
     - Istantanea
     - Backup
     - Ripristinare
     - Failover
   - **spec.stage**: (*Obbligatorio*) Una stringa che indica in quale fase dell'azione deve essere eseguito l'hook di esecuzione. Questa serie di hook non prevede hook in nessun'altra fase. Valori possibili:
     - Pre
     - Inviare

     Esempio YAML:

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

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

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

Utilizzare la CLI

Passi

1. Creare la richiesta di esecuzione manuale dell'hook:

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

2. Controllare lo stato dell'esecuzione del gancio. È possibile eseguire questo comando più volte fino al completamento dell'operazione:

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

3. Descrivi l'oggetto exechooksrun per vedere i dettagli finali e lo stato:

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

Informazioni sul copyright

Informazioni sul marchio commerciale