Gestire e proteggere le applicazioni

Trident

NetApp
April 08, 2026

Sommario

Gestire e proteggere le applicazioni

Utilizzare gli oggetti Trident Protect AppVault per gestire i bucket

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

È necessario creare una AppVault CR manualmente o dalla riga di comando quando si eseguono operazioni di protezione dei dati su un'applicazione. La AppVault CR è specifica per il proprio ambiente e puoi utilizzare gli esempi in questa pagina come guida quando crei AppVault CR.

> Assicurarsi che la AppVault CR sia presente sul cluster in cui è installato Trident Protect. Se la AppVault CR non esiste o non è possibile accedervi, la riga di comando mostra un errore.

Configura l'autenticazione e le password di AppVault

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

Password del repository del data mover

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

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

Crea un secret per la password del repository del data mover

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

> - A seconda del data mover utilizzato, è sufficiente includere la password corrispondente per quel data mover. Ad esempio, se si utilizza Restic e non si prevede di utilizzare Kopia in futuro, è possibile includere solo la password Restic quando si crea il segreto.
> - Conserva la password in un luogo sicuro. Ti servirà per ripristinare i dati sullo stesso cluster o su uno diverso. Se il cluster o il `trident-protect` namespace viene eliminato, non potrai ripristinare i backup o gli snapshot senza la password.

Utilizzare un CR

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

Usa la CLI

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

Autorizzazioni IAM di storage compatibile S3

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

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

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

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

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

Requisiti per EKS Pod Identity con Trident Protect

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

- Il tuo cluster EKS ha abilitato Pod Identity.
- Hai creato un ruolo IAM con le necessarie autorizzazioni per il bucket S3. Per saperne di più, consulta "Autorizzazioni IAM di storage compatibile S3".
- Il ruolo IAM è associato ai seguenti account del servizio Trident Protect:
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

Per istruzioni dettagliate sull'abilitazione di Pod Identity e sull'associazione dei ruoli IAM agli account di servizio, consultare la "Documentazione AWS EKS Pod Identity".

AppVault Configuration Quando si usa EKS Pod Identity, configura il tuo AppVault CR con il flag `useIAM: true` invece di credenziali esplicite:

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

AppVault esempi di generazione chiave per cloud provider

Quando si definisce un AppVault CR, è necessario includere le credenziali per accedere alle risorse ospitate dal provider, a meno che non si utilizzi IAM authentication. Le modalità di generazione delle chiavi per le credenziali variano a seconda del provider. Di seguito sono riportati esempi di generazione di chiavi da riga di comando per diversi provider. È possibile utilizzare i seguenti esempi per creare chiavi per le credenziali di ciascun cloud provider.

Google Cloud

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

Amazon S3 (AWS)

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

Microsoft Azure

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

S3 generico

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

ONTAP S3

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

StorageGrid S3

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

Esempi di creazione di AppVault

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

AppVault esempi di CR

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

- È possibile specificare facoltativamente un segreto Kubernetes che contiene password personalizzate per la crittografia dei repository Restic e Kopia. Fare riferimento a Password del repository del data mover per ulteriori informazioni.
- Per gli oggetti Amazon S3 (AWS) AppVault, puoi specificare facoltativamente un sessionToken, che è utile se utilizzi il single sign-on (SSO) per l'autenticazione. Questo token viene creato quando generi le chiavi per il provider in AppVault esempi di generazione chiave per cloud provider.
- Per gli oggetti S3 AppVault, puoi facoltativamente specificare un URL proxy di egress per il traffico S3 outbound utilizzando la chiave `spec.providerConfig.S3.proxyURL`.

Google Cloud

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

Amazon S3 (AWS)

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

Per gli ambienti EKS che utilizzano Pod Identity con Kopia data mover, puoi rimuovere la sezione `providerCredentials` e aggiungere `useIAM: true` sotto la configurazione `s3`.

Microsoft Azure

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

S3 generico

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

ONTAP S3

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

StorageGrid S3

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

Esempi di creazione di AppVault utilizzando la CLI di Trident Protect

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

- È possibile specificare facoltativamente un segreto Kubernetes che contiene password personalizzate per la crittografia dei repository Restic e Kopia. Fare riferimento a Password del repository del data mover per ulteriori informazioni.
- Per gli oggetti S3 AppVault, puoi facoltativamente specificare un URL proxy di egress per il traffico S3 outbound utilizzando l'argomento `--proxy-url <ip_address:port>`.

Google Cloud

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

Amazon S3 (AWS)

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

Microsoft Azure

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

S3 generico

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

ONTAP S3

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

StorageGrid S3

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

Opzioni di configurazione `providerConfig.s3` supportate

Consultare la tabella seguente per le opzioni di configurazione del provider S3:

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

Visualizza informazioni AppVault

È possibile utilizzare il plugin Trident Protect CLI per visualizzare le informazioni sugli oggetti AppVault che hai creato sul cluster.

Passaggi

1. Visualizza il contenuto di un AppVault object:

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

Esempio output:

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

2. Opzionalmente, per vedere il AppVaultPath per ogni risorsa, usa il flag `--show-paths`.

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

Rimuovi un AppVault

È possibile rimuovere un AppVault oggetto in qualsiasi momento.

Non rimuovere la `finalizers` chiave nel AppVault CR prima di eliminare l'oggetto AppVault. In caso contrario, potrebbero rimanere dati residui nel bucket AppVault e risorse orfane nel cluster.

Prima di iniziare

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

Rimuovi un AppVault utilizzando la CLI di Kubernetes

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

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

Rimuovi un AppVault utilizzando la Trident Protect CLI

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

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

Definisci un'applicazione per la gestione con Trident Protect

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

Crea una AppVault CR

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

Definire un'applicazione

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

Aggiungi un'applicazione utilizzando un CR

Passaggi

1. Crea il file CR dell'applicazione di destinazione:
   a. Creare il file custom resource (CR) e assegnargli un nome (ad esempio `maria-app.yaml`).
   b. Configura i seguenti attributi:
      - **metadata.name**: (*Obbligatorio*) Il nome della risorsa personalizzata dell'applicazione. Nota il nome che scegli perché altri file CR necessari per le operazioni di protezione fanno riferimento a questo valore.
      - **spec.includedNamespaces**: (*Obbligatorio*) Utilizzare il selettore di namespace e di etichetta per specificare i namespace e le risorse che l'applicazione utilizza. Il namespace dell'applicazione deve essere parte di questo elenco. Il selettore di etichetta è facoltativo e può essere utilizzato per filtrare le risorse all'interno di ciascun namespace specificato.
      - **spec.includedClusterScopedResources**: (*Facoltativo*) Utilizzare questo attributo per specificare le risorse con ambito cluster da includere nella definizione dell'applicazione. Questo attributo consente di selezionare queste risorse in base al gruppo, alla versione, al tipo e alle etichette.
        - **groupVersionKind**: (*Obbligatorio*) Specifica il gruppo API, la versione e il tipo di risorsa con ambito cluster.
        - **labelSelector**: (*Facoltativo*) Filtra le risorse con ambito cluster in base alle loro etichette.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Facoltativo*) Questa annotazione è applicabile solo alle applicazioni definite da macchine virtuali, ad esempio in ambienti KubeVirt, in cui si verificano blocchi del file system prima degli snapshot. Specificare se l'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 "impostazione globale di congelamento Trident Protect".

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

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

+
Esempio YAML:

+

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

1. (*Facoltativo*) Aggiungi il filtraggio che include o esclude 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 resourceMatcher oggetti. Se si definiscono più elementi in questo array, la corrispondenza avviene tramite un'operazione OR, e i campi all'interno di ciascun elemento (group, kind, version) corrispondono tramite un'operazione AND.
       - **resourceMatchers[].group**: (*Facoltativo*) Gruppo della risorsa da filtrare.
       - **resourceMatchers[].kind**: (*Facoltativo*) Tipo di risorsa da filtrare.
       - **resourceMatchers[].version**: (*Facoltativo*) Versione della risorsa da filtrare.

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

> Quando sia `resourceFilter` che `labelSelector` vengono utilizzati, `resourceFilter` viene eseguito per primo e poi `labelSelector` viene applicato alle risorse risultanti.

Ad esempio:

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

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

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

Passaggi

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

   Facoltativamente, puoi utilizzare un'annotazione quando crei un'app per specificare se l'applicazione può scrivere sul filesystem durante uno snapshot. Questo è applicabile solo alle applicazioni definite da macchine virtuali, come negli ambienti KubeVirt, dove il filesystem viene bloccato prima degli snapshot. Se imposti l'annotazione su `true`, l'applicazione ignora l'impostazione globale e può scrivere sul filesystem durante uno snapshot. Se la imposti su `false`, l'applicazione ignora

l'impostazione globale e il filesystem viene bloccato durante uno snapshot. Se utilizzi l'annotazione ma l'applicazione non ha macchine virtuali nella definizione dell'applicazione, l'annotazione viene ignorata. Se non utilizzi l'annotazione, l'applicazione segue "impostazione globale di congelamento Trident Protect".

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

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

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

- Crea l'applicazione e configura l'impostazione dell'applicazione locale per il comportamento di freeze del filesystem:

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

È possibile utilizzare `--resource-filter-include` e `--resource-filter-exclude` flag per includere o escludere risorse in base a `resourceSelectionCriteria` come gruppo, tipo, versione, etichette, nomi e namespace, come mostrato nell'esempio seguente:

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

Proteggi le applicazioni utilizzando Trident Protect

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

È possibile configurare Trident Protect per congelare e scongelare i filesystem durante le operazioni di protezione dei dati. "Scopri di più sulla configurazione del congelamento del filesystem con Trident Protect".

Crea un'istantanea su richiesta

È possibile creare una snapshot su richiesta in qualsiasi momento.

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

Crea un'istantanea utilizzando un CR

Passaggi

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

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

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

Crea un'istantanea utilizzando la CLI

Passaggi

1. Crea l'istantanea, sostituendo i valori tra parentesi con le informazioni del tuo ambiente. Ad esempio:

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

Crea un backup su richiesta

È possibile eseguire il backup di un'app in qualsiasi momento.

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

Prima di iniziare

Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di backup s3 di lunga durata. Se il token scade durante l'operazione di backup, l'operazione può fallire.

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

Crea un backup utilizzando un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-backup-cr.yaml`.
2. Nel file che hai creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.applicationRef**: (*Obbligatorio*) Il nome Kubernetes dell'applicazione di cui eseguire il backup.
   - **spec.appVaultRef**: (*Required*) Il nome del AppVault in cui devono essere archiviati i contenuti del backup.
   - **spec.dataMover**: (*Opzionale*) Una stringa che indica quale strumento di backup utilizzare per l'operazione di backup. Valori possibili (case sensitive):
     - `Restic`
     - `Kopia` (predefinito)
   - **spec.reclaimPolicy**: (*Opzionale*) Definisce cosa succede a un backup quando viene rilasciato dalla sua richiesta. Valori possibili:
     - `Delete`
     - `Retain` (predefinito)
   - **spec.snapshotRef**: (*Opzionale*): Nome dell'istantanea da usare come origine del backup. Se non fornito, verrà creata un'istantanea temporanea e verrà eseguito il backup.

     Esempio YAML:

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

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

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

Crea un backup utilizzando la CLI

Passaggi

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

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

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

Annotazioni di backup supportate

La tabella seguente descrive le annotazioni che puoi utilizzare quando crei un backup CR:

Annotazione	Tipo	Descrizione	Valore predefinito
protect.trident.netapp.io/full-backup	stringa	Specifica se un backup deve essere non incrementale. Impostare su `true` per creare un backup non incrementale. È best practice eseguire periodicamente un backup completo e poi eseguire backup incrementali tra un backup completo e l'altro per ridurre al minimo il rischio associato ai ripristini.	"false"
protect.trident.netapp.io/snapshot-completion-timeout	stringa	Il tempo massimo consentito per il completamento dell'operazione di snapshot complessiva.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	stringa	Tempo massimo consentito affinché gli snapshot del volume raggiungano lo stato pronto all'uso.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	stringa	Il tempo massimo consentito per la creazione di snapshot del volume.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	stringa	Tempo massimo (in secondi) di attesa affinché eventuali nuove PersistentVolumeClaims (PVC) raggiungano la fase `Bound` prima che l'operazione fallisca.	"1200" (20 minuti)

Crea una pianificazione della protezione dei dati

Una policy di protezione protegge un'app creando snapshot, backup o entrambi secondo una pianificazione definita. Puoi scegliere di creare snapshot e backup ogni ora, ogni giorno, ogni settimana e ogni mese, e puoi specificare il numero di copie da conservare. Puoi pianificare un backup completo non incrementale utilizzando l'annotazione full-backup-rule. Per impostazione predefinita, tutti i backup sono incrementali. Eseguire periodicamente un backup completo, insieme a backup incrementali tra un backup completo e l'altro, aiuta a ridurre il rischio associato ai ripristini.

- È possibile creare pianificazioni per le istantanee solo impostando `backupRetention` su zero e `snapshotRetention` su un valore maggiore di zero. Impostare `snapshotRetention` su zero significa che qualsiasi backup pianificato creerà comunque istantanee, ma queste sono temporanee e vengono eliminate immediatamente dopo il completamento del backup.
- Le risorse con ambito cluster vengono incluse in un backup, snapshot o clone se sono esplicitamente referenziate nella definizione dell'applicazione o se hanno riferimenti a uno qualsiasi degli spazi dei nomi dell'applicazione.

Crea una pianificazione utilizzando un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-schedule-cr.yaml`.
2. Nel file che hai creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.dataMover**: (*Opzionale*) Una stringa che indica quale strumento di backup utilizzare per l'operazione di backup. Valori possibili (case sensitive):
     - `Restic`
     - `Kopia` (predefinito)
   - **spec.applicationRef**: il nome Kubernetes dell'applicazione di cui eseguire il backup.
   - **spec.appVaultRef**: (*Required*) Il nome del AppVault in cui devono essere archiviati i contenuti del backup.
   - **spec.backupRetention**: (*Required*) Il numero di backup da conservare. Zero indica che non devono essere creati backup (solo istantanee).
   - **backupReclaimPolicy**: (*Opzionale*) Determina cosa succede a un backup se il CR di backup viene eliminato durante il suo periodo di conservazione. Dopo il periodo di conservazione, i backup vengono sempre eliminati. Valori possibili (case sensitive):
     - `Retain` (predefinito)
     - `Delete`
   - **spec.snapshotRetention**: (*Required*) Il numero di istantanee da conservare. Zero indica che non devono essere create istantanee.
   - **snapshotReclaimPolicy**: (*Opzionale*) Determina cosa succede a una snapshot se la CR della snapshot viene eliminata durante il suo periodo di conservazione. Dopo il periodo di conservazione, le snapshot vengono sempre eliminate. Valori possibili (case sensitive):
     - `Retain`
     - `Delete` (predefinito)
   - **spec.granularity**: La frequenza con cui la pianificazione deve essere eseguita. Valori possibili, con i campi associati richiesti:
     - `Hourly` (richiede che tu specifichi `spec.minute`)
     - `Daily` (richiede che tu specifichi `spec.minute` e `spec.hour`)
     - `Weekly` (richiede che tu specifichi `spec.minute, spec.hour`, e `spec.dayOfWeek`)
     - `Monthly` (richiede che tu specifichi `spec.minute, spec.hour`, e `spec.dayOfMonth`)
     - `Custom`
   - **spec.dayOfMonth**: (*Facoltativo*) Il giorno del mese (1 - 31) in cui deve essere eseguita la pianificazione. Questo campo è obbligatorio se la granularità è impostata su `Monthly`. Il valore deve essere fornito come stringa.
   - **spec.dayOfWeek**: (*Facoltativo*) Il giorno della settimana (0 - 7) in cui deve essere eseguita la pianificazione. I valori 0 o 7 indicano la domenica. Questo campo è obbligatorio se la granularità è impostata su `Weekly`. Il valore deve essere fornito come stringa.

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

  Esempio YAML per la pianificazione di backup e snapshot:

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

  Esempio di YAML per la pianificazione solo snapshot:

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

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

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

Crea una pianificazione utilizzando la CLI

Passaggi

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

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

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

   I seguenti flag forniscono un controllo aggiuntivo sulla tua pianificazione:

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

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

   - **Pianificazioni solo snapshot**: imposta `--backup-retention 0` e specifica un valore maggiore di zero per `--snapshot-retention`.

Annotazioni di pianificazione supportate

La tabella seguente descrive le annotazioni che è possibile utilizzare quando si crea una schedule CR:

Annotazione	Tipo	Descrizione	Valore predefinito
protect.trident.netapp.io/full-backup-rule	stringa	Specifica la regola per la pianificazione dei backup completi. Puoi impostarla su `Always` per backup completi costanti o personalizzarla in base alle tue esigenze. Ad esempio, se scegli la granularità giornaliera, puoi specificare i giorni della settimana in cui deve essere eseguito il backup completo (ad esempio, `"Monday,Thursday"`). I valori validi per i giorni della settimana sono: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday. Nota che questa annotazione può essere utilizzata solo con pianificazioni che hanno `granularity` impostato su `Daily`.	Non impostato (tutti i backup sono incrementali)
protect.trident.netapp.io/snapshot-completion-timeout	stringa	Il tempo massimo consentito per il completamento dell'operazione di snapshot complessiva.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	stringa	Tempo massimo consentito affinché gli snapshot del volume raggiungano lo stato pronto all'uso.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	stringa	Il tempo massimo consentito per la creazione di snapshot del volume.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	stringa	Tempo massimo (in secondi) di attesa affinché eventuali nuove PersistentVolumeClaims (PVC) raggiungano la fase `Bound` prima che l'operazione fallisca.	"1200" (20 minuti)

Elimina una snapshot

Elimina gli snapshot pianificati o on-demand di cui non hai più bisogno.

Passaggi

1. Rimuovi lo snapshot CR associato allo snapshot:

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

Elimina un backup

Elimina i backup pianificati o on-demand di cui non hai più bisogno.

Assicurarsi che la policy di reclaim sia impostata su `Delete` per rimuovere tutti i dati di backup dallo storage a oggetti. L'impostazione predefinita della policy è `Retain` per evitare la perdita accidentale di dati. Se la policy non viene modificata su `Delete`, i dati di backup rimarranno nello storage a oggetti e richiederanno l'eliminazione manuale.

Passaggi

1. Rimuovi il CR di backup associato al backup:

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

Verificare lo stato di un'operazione di backup

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

Passaggi

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

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

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

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

Prima di iniziare

Assicurarsi quanto segue:

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

Espandi per i passaggi di configurazione

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

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

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

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

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

   Risposta:

   ```
   snapshotDirectory: "true"
   ```

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

Risultato

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

Ripristina le applicazioni

Ripristina le applicazioni utilizzando Trident Protect

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

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

Ripristina da un backup a un namespace diverso

Quando si ripristina un backup in un namespace diverso utilizzando una BackupRestore CR, Trident Protect ripristina l'applicazione in un nuovo namespace e crea una application CR per l'applicazione ripristinata. Per proteggere l'applicazione ripristinata, crea backup o snapshot on-demand oppure stabilisci una pianificazione di protezione.

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

Prima di iniziare

Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino s3 di lunga durata. Se il token scade durante l'operazione di ripristino, l'operazione può fallire.

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

Quando si ripristinano i backup utilizzando Kopia come data mover, è possibile specificare facoltativamente annotazioni nel CR o tramite la CLI per controllare il comportamento dello storage temporaneo utilizzato da Kopia. Consultare il "Documentazione Kopia" per ulteriori informazioni sulle opzioni che è possibile configurare. Utilizzare il `tridentctl-protect create --help` comando per ulteriori informazioni sulla specifica delle annotazioni con la Trident Protect CLI.

Utilizzare un CR

Passaggi

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

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

   - **spec.appVaultRef**: (*Obbligatorio*) Il nome del AppVault in cui sono archiviati i contenuti del backup.
   - **spec.namespaceMapping**: Il mapping dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituisci `my-source-namespace` e `my-destination-namespace` con le informazioni del tuo ambiente.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: BackupRestore
     metadata:
       name: my-cr-name
       namespace: my-destination-namespace
     spec:
       appArchivePath: my-backup-path
       appVaultRef: appvault-name
       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 che selezioni. Ad esempio, se selezioni una risorsa di richiesta di volume persistente e questa ha un pod associato, Trident Protect ripristinerà anche il pod associato.

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

all'interno di ciascun elemento (group, kind, version) corrispondono tramite un'operazione AND.

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

  Ad esempio:

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

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

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

Usa la CLI

Passaggi

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

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

Ripristina da un backup nello spazio dei nomi originale

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

Prima di iniziare

Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino s3 di lunga durata. Se il token scade durante l'operazione di ripristino, l'operazione può fallire.

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

> Quando si ripristinano i backup utilizzando Kopia come data mover, è possibile specificare facoltativamente annotazioni nel CR o tramite la CLI per controllare il comportamento dello storage temporaneo utilizzato da Kopia. Consultare il "Documentazione Kopia" per ulteriori informazioni sulle opzioni che è possibile configurare. Utilizzare il `tridentctl-protect create --help` comando per ulteriori informazioni sulla specifica delle annotazioni con la Trident Protect CLI.

Utilizzare un CR

Passaggi

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

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

   - **spec.appVaultRef**: (*Obbligatorio*) Il nome del AppVault in cui sono archiviati i contenuti del backup.

     Ad esempio:

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

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

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

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

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

  Ad esempio:

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

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

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

Usa la CLI

Passaggi

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

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

Ripristina da un backup a un cluster diverso

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

- Quando si ripristinano i backup utilizzando Kopia come data mover, è possibile specificare facoltativamente annotazioni nel CR o tramite la CLI per controllare il comportamento dello storage temporaneo utilizzato da Kopia. Consultare il "Documentazione Kopia" per ulteriori informazioni sulle opzioni che è possibile configurare. Utilizzare il `tridentctl-protect create --help` comando per ulteriori informazioni sulla specifica delle annotazioni con la Trident Protect CLI.
- Quando si utilizza una CR per ripristinare in un nuovo namespace, è necessario creare manualmente il namespace di destinazione prima di applicare la CR. Trident Protect crea automaticamente i namespace solo quando si utilizza la CLI.

Prima di iniziare

Assicurarsi che siano soddisfatti i seguenti prerequisiti:

- Il cluster di destinazione ha Trident Protect installato.
- Il cluster di destinazione ha accesso al percorso del bucket dello stesso AppVault del cluster di origine, dove è archiviato il backup.
- Assicurati che l'ambiente locale possa connettersi al bucket di storage a oggetti definito nella AppVault CR durante l'esecuzione del comando `tridentctl-protect get appvaultcontent`. Se le restrizioni di rete impediscono l'accesso, esegui la CLI di Trident Protect da un pod sul cluster di destinazione invece.
- Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino di lunga durata. Se il token scade durante l'operazione di ripristino, l'operazione può fallire.
  - Fare riferimento a "Documentazione API AWS" per ulteriori informazioni sulla verifica della scadenza del token della sessione corrente.
  - Fare riferimento a "Documentazione AWS" per ulteriori informazioni sulle credenziali con le risorse AWS.

Passaggi

1. Verifica che la AppVault CR esista sul cluster di destinazione utilizzando il plugin CLI di Trident Protect:

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

   Se il AppVault CR non esiste sul cluster di destinazione, crealo seguendo i passaggi in "Utilizzare gli oggetti Trident Protect AppVault per gestire i bucket".

2. Visualizza il contenuto del backup disponibile AppVault sul cluster di destinazione e prendi nota `appArchivePath` del backup che desideri ripristinare:

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

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

Esempio di output:

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

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

> Quando si utilizza una CR, assicurarsi che lo spazio dei nomi destinato al ripristino dell’applicazione esista sul cluster di destinazione.

Utilizzare un CR

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-backup-restore-cr.yaml`.
2. Nel file che hai creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.appVaultRef**: (*Obbligatorio*) Il nome del AppVault in cui sono archiviati i contenuti del backup.
   - **spec.appArchivePath**: (*Obbligatorio*) Il percorso all'interno di AppVault in cui sono archiviati i contenuti del backup. Utilizzare il comando del passaggio 2 per visualizzare i contenuti del backup e trovare `appArchivePath` per il backup che si desidera ripristinare.
   - **spec.namespaceMapping**: Il mapping dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituisci `my-source-namespace` e `my-destination-namespace` con le informazioni del tuo ambiente.

     Ad esempio:

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

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

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

Usa la CLI

1. Utilizzare il seguente comando per ripristinare l'applicazione, sostituendo i valori tra parentesi con le informazioni del proprio ambiente. L'argomento namespace-mapping utilizza namespace separati da due punti per mappare i namespace di origine ai namespace di destinazione corretti nel formato source1:dest1,source2:dest2. Ad esempio:

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

Ripristina da uno Snapshot a uno spazio dei nomi diverso

È possibile ripristinare i dati da uno snapshot utilizzando un file di risorsa personalizzata (CR) sia in un namespace diverso che nel namespace di origine. Quando si ripristina uno snapshot in un namespace diverso utilizzando una SnapshotRestore CR, Trident Protect ripristina l'applicazione in un nuovo namespace e crea una CR dell'applicazione per l'applicazione ripristinata. Per proteggere l'applicazione ripristinata, crea backup o snapshot on-demand oppure stabilisci una pianificazione di protezione.

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

Prima di iniziare

Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino s3 di lunga durata. Se il token scade durante l'operazione di ripristino, l'operazione può fallire.

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

Utilizzare un CR

Passaggi

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

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

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

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

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

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

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

all'interno di ciascun elemento (group, kind, version) corrispondono tramite un'operazione AND.

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

  Ad esempio:

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

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

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

Usa la CLI

Passaggi

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

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

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

  Ad esempio:

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

Ripristina da uno Snapshot allo spazio dei nomi originale

È possibile ripristinare uno snapshot nel namespace originale in qualsiasi momento.

> Se la tua applicazione utilizza più namespace e questi namespace hanno PVC con lo stesso nome, le operazioni di ripristino snapshot (sia sul posto che in un nuovo namespace) non funzioneranno correttamente. Tutti i volumi ripristinati avranno gli stessi dati invece dei dati corretti per ciascun namespace. Utilizza il ripristino da backup invece del ripristino da snapshot, oppure aggiorna alla versione 26.02 o successiva che risolve questo problema.

Prima di iniziare

Assicurarsi che la scadenza del token di sessione AWS sia sufficiente per qualsiasi operazione di ripristino s3 di lunga durata. Se il token scade durante l'operazione di ripristino, l'operazione può fallire.

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

Utilizzare un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-snapshot-ipr-cr.yaml`.
2. Nel file che hai creato, configura i seguenti attributi:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.appVaultRef**: (*Obbligatorio*) Il nome del AppVault in cui sono archiviati i contenuti dello snapshot.
   - **spec.appArchivePath**: Il percorso all'interno di AppVault in cui sono archiviati i contenuti dello snapshot. È possibile utilizzare il seguente comando per trovare questo percorso:

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

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

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

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

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

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

  Ad esempio:

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

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

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

Usa la CLI

Passaggi

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

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

Verificare lo stato di un'operazione di ripristino

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

Passaggi

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

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

Utilizza le impostazioni di ripristino avanzate di Trident Protect

È possibile personalizzare le operazioni di ripristino utilizzando impostazioni avanzate come annotazioni, impostazioni dello spazio dei nomi e opzioni di storage per soddisfare i requisiti specifici.

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

Durante le operazioni di ripristino e failover, le etichette e le annotazioni nello spazio dei nomi di destinazione vengono rese corrispondenti alle etichette e alle annotazioni nello spazio dei nomi di origine. Le etichette o le annotazioni dello spazio dei nomi di origine che non esistono nello spazio dei nomi di destinazione vengono aggiunte e tutte le etichette o annotazioni già esistenti vengono sovrascritte per corrispondere al valore dello spazio dei nomi di origine. Le etichette o le annotazioni che esistono solo nello spazio dei nomi di destinazione rimangono invariate.

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

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

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

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 escluse dall'operazione di ripristino o failover. Assicurarsi che queste impostazioni siano configurate durante l'installazione iniziale di Helm. Per ulteriori informazioni, consultare "Configura impostazioni aggiuntive dell'helm chart Trident Protect".

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

Esempio

Il seguente esempio presenta uno spazio dei nomi di origine e uno di destinazione, ciascuno con annotazioni ed etichette diverse. Puoi vedere lo stato dello spazio dei nomi di destinazione prima e dopo l'operazione e come le annotazioni e le etichette vengono combinate o sovrascritte nello spazio dei nomi di destinazione.

Prima dell'operazione di ripristino o failover

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

Spazio dei nomi	Annotazioni	Etichette
Namespace ns-1 (origine)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• ambiente=produzione • compliance=hipaa • name=ns-1
Namespace ns-2 (destinazione)	• annotation.one/key: "true" • annotazione.tre/chiave: "false"	• role=database

Dopo l'operazione di ripristino

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

Spazio dei nomi	Annotazioni	Etichette
Namespace ns-2 (destinazione)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotazione.tre/chiave: "false"	• name=ns-2 • compliance=hipaa • ambiente=produzione • role=database

Campi supportati

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

Mappatura delle storage class

L' `spec.storageClassMapping`attributo definisce una mappatura da una classe di storage presente nell'applicazione di origine a una nuova classe di storage nel cluster di destinazione. Puoi utilizzare questa opzione quando migri applicazioni tra cluster con classi di storage diverse o quando cambi il backend di storage per le operazioni di BackupRestore.

Esempio:

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

Annotazioni supportate

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

Annotazione	Tipo	Descrizione	Valore predefinito
protect.trident.netapp.io/data-mover-timeout-sec	stringa	Il tempo massimo (in secondi) consentito per l'operazione di spostamento dei dati che può essere bloccata.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	stringa	Limite massimo di dimensione (in megabyte) per la cache dei contenuti Kopia.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	stringa	Tempo massimo (in secondi) di attesa affinché eventuali nuove PersistentVolumeClaims (PVC) raggiungano la fase `Bound` prima che l'operazione fallisca. Si applica a tutti i tipi di restore CR (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Utilizzare un valore più alto se il backend di storage o il cluster richiede spesso più tempo.	"1200" (20 minuti)

Replicare le applicazioni utilizzando NetApp SnapMirror e Trident Protect

Utilizzando Trident Protect, puoi utilizzare le capacità di replica asincrona della tecnologia NetApp SnapMirror per replicare dati e modifiche delle applicazioni da un backend di

storage a un altro, sullo stesso cluster o tra cluster diversi.

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

Durante le operazioni di ripristino e failover, le etichette e le annotazioni nello spazio dei nomi di destinazione vengono rese corrispondenti alle etichette e alle annotazioni nello spazio dei nomi di origine. Le etichette o le annotazioni dello spazio dei nomi di origine che non esistono nello spazio dei nomi di destinazione vengono aggiunte e tutte le etichette o annotazioni già esistenti vengono sovrascritte per corrispondere al valore dello spazio dei nomi di origine. Le etichette o le annotazioni che esistono solo nello spazio dei nomi di destinazione rimangono invariate.

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

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

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

> 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 escluse dall'operazione di ripristino o failover. Assicurarsi che queste impostazioni siano configurate durante l'installazione iniziale di Helm. Per ulteriori informazioni, consultare "Configura impostazioni aggiuntive dell'helm chart Trident Protect".

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

Esempio

Il seguente esempio presenta uno spazio dei nomi di origine e uno di destinazione, ciascuno con annotazioni ed etichette diverse. Puoi vedere lo stato dello spazio dei nomi di destinazione prima e dopo l'operazione e come le annotazioni e le etichette vengono combinate o sovrascritte nello spazio dei nomi di destinazione.

Prima dell'operazione di ripristino o failover

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

Spazio dei nomi	Annotazioni	Etichette
Namespace ns-1 (origine)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• ambiente=produzione • compliance=hipaa • name=ns-1
Namespace ns-2 (destinazione)	• annotation.one/key: "true" • annotazione.tre/chiave: "false"	• role=database

Dopo l'operazione di ripristino

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

Spazio dei nomi	Annotazioni	Etichette
Namespace ns-2 (destinazione)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotazione.tre/chiave: "false"	• name=ns-2 • compliance=hipaa • ambiente=produzione • role=database

È possibile configurare Trident Protect per congelare e scongelare i filesystem durante le operazioni di protezione dei dati. "Scopri di più sulla configurazione del congelamento del filesystem con Trident Protect".

Hook di esecuzione durante le operazioni di failover e reverse

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

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

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

Imposta una relazione di replica

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

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

Passaggi

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

Crea un AppVault utilizzando un CR

a. Creare il file custom resource (CR) e assegnargli un nome (ad esempio `trident-protect-appvault-primary-source.yaml`).

b. Configura i seguenti attributi:

- **metadata.name**: (*Obbligatorio*) Il nome della risorsa personalizzata AppVault. Prendi nota del nome che scegli, perché altri file CR necessari per una relazione di replica fanno riferimento a questo valore.
- **spec.providerConfig**: (*Obbligatorio*) Memorizza la configurazione necessaria per accedere a AppVault utilizzando il provider specificato. Scegli un bucketName e qualsiasi altro dettaglio necessario per il tuo provider. Prendi nota dei valori che scegli, perché altri file CR necessari per una relazione di replica fanno riferimento a questi valori. Fai riferimento a "Risorse personalizzate AppVault" per esempi di CR AppVault con altri provider.
- **spec.providerCredentials**: (*Obbligatorio*) Memorizza i riferimenti a qualsiasi credenziale richiesta per accedere al AppVault utilizzando il provider specificato.
  - **spec.providerCredentials.valueFromSecret**: (*Obbligatorio*) Indica che il valore delle credenziali deve provenire da un segreto.
    - **key**: (*Obbligatorio*) La chiave valida del segreto da selezionare.
    - **name**: (*Obbligatorio*) Nome del secret contenente il valore per questo campo. Deve essere nello stesso namespace.
  - **spec.providerCredentials.secretAccessKey**: (*Obbligatorio*) La chiave di accesso utilizzata per accedere al provider. Il **nome** deve corrispondere a **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*Obbligatorio*) Determina cosa fornisce il backup; ad esempio, NetApp ONTAP S3, S3 generico, Google Cloud o Microsoft Azure. Valori possibili:
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

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

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

Creare un AppVault utilizzando la CLI

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

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

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

Crea l'applicazione di origine utilizzando un CR

a. Creare il file custom resource (CR) e assegnargli un nome (ad esempio `trident-protect-app-source.yaml`).

b. Configura i seguenti attributi:

- **metadata.name**: (*Required*) Il nome della risorsa personalizzata dell'applicazione. Prendi nota del nome che scegli, perché altri file CR necessari per una relazione di replica fanno riferimento a questo valore.
- **spec.includedNamespaces**: (*Required*) Un array di spazi dei nomi e di etichette associate. Usare i nomi degli spazi dei nomi e, facoltativamente, restringere l'ambito degli spazi dei nomi con le etichette per specificare le risorse che esistono negli spazi dei nomi qui elencati. Lo spazio dei nomi dell'applicazione deve far parte di questo array.

  Esempio YAML:

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

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

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

Crea l'applicazione sorgente utilizzando la CLI

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

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

3. Facoltativamente, sul cluster di origine, acquisire un'istantanea dell'applicazione di origine. Questa istantanea viene utilizzata come base per l'applicazione sul cluster di destinazione. Se si salta questo passaggio, sarà necessario attendere l'esecuzione della prossima istantanea programmata per avere un'istantanea recente. Per creare un'istantanea su richiesta, fare riferimento a "Crea un'istantanea su richiesta".

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

Oltre alla pianificazione fornita di seguito, si consiglia di creare una pianificazione separata delle snapshot giornaliere con un periodo di conservazione di 7 giorni per mantenere una snapshot comune tra i cluster ONTAP peered. Questo garantisce che le snapshot siano disponibili per un massimo di 7 giorni, ma il periodo di conservazione può essere personalizzato in base alle esigenze degli utenti.

Se si verifica un failover, il sistema può utilizzare queste snapshot per un massimo di 7 giorni per le operazioni di inversione. Questo approccio rende il processo di inversione più rapido ed efficiente perché vengono trasferite solo le modifiche apportate dall'ultima snapshot, non tutti i dati.

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

Crea il programma di replica utilizzando un CR

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

   i. Creare il file custom resource (CR) e assegnargli un nome (ad esempio `trident-protect-schedule.yaml`).

   ii. Configura i seguenti attributi:

   - **metadata.name**: (*Richiesta*) Il nome della risorsa personalizzata della pianificazione.
   - **spec.appVaultRef**: (*Required*) Questo valore deve corrispondere al campo metadata.name di AppVault per l'applicazione di origine.
   - **spec.applicationRef**: (*Required*) Questo valore deve corrispondere al campo metadata.name dell'applicazione CR di origine.
   - **spec.backupRetention**: (*Required*) Questo campo è obbligatorio e il valore deve essere impostato su 0.
   - **spec.enabled**: Deve essere impostato su true.
   - **spec.granularity**: Deve essere impostato su `Custom`.
   - **spec.recurrenceRule**: Definire una data di inizio in ora UTC e un intervallo di ricorrenza.
   - **spec.snapshotRetention**: Deve essere impostato su 2.

     Esempio YAML:

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

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

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

Crea il programma di replica utilizzando la CLI

a. Crea il programma di replica, sostituendo i valori tra parentesi con le informazioni del tuo ambiente:

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

Esempio:

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

5. Sul cluster di destinazione, crea una source application AppVault CR identica alla AppVault CR che hai applicato sul cluster di origine e assegnale un nome (ad esempio, `trident-protect-appvault-primary-destination.yaml`).
6. Applica il CR:

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

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

credenziali deve provenire da un segreto.

- **key**: (*Obbligatorio*) La chiave valida del segreto da selezionare.
- **name**: (*Obbligatorio*) Nome del secret contenente il valore per questo campo. Deve essere nello stesso namespace.

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

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

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

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

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

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

Crea un AppMirrorRelationship utilizzando un CR

a. Creare il file custom resource (CR) e assegnargli un nome (ad esempio `trident-protect-relationship.yaml`).

b. Configura i seguenti attributi:

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

  Esempio YAML:

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

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

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

Creare un AppMirrorRelationship utilizzando la CLI

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

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

Esempio:

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

9. (*Opzionale*) Sul cluster di destinazione, verificare lo stato e la situazione della relazione di replica:

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

Fail over verso il cluster di destinazione

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

Passaggi

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

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

4. (*Opzionale*) Crea tutte le policy di protezione di cui hai bisogno sull'applicazione su cui è stato effettuato il failover.
5. (*Opzionale*) Controlla lo stato e lo stato della relazione di replica:

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

Risincronizza una relazione di replica sottoposta a failover

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

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

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

Passaggi

1. Opzionale: Sul cluster di origine, crea uno snapshot dell'applicazione di origine. Questo assicura che le ultime modifiche dal cluster di origine vengano acquisite.
2. Sul cluster di destinazione, modificare il file CR AppMirrorRelationship (ad esempio, `trident-protect-relationship.yaml`) e cambiare il valore di spec.desiredState in `Established`.
3. Salva il file CR.
4. Applica il CR:

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

5. Se sono state create pianificazioni di protezione sul cluster di destinazione per proteggere l'applicazione sottoposta a failover, rimuoverle. Eventuali pianificazioni rimaste causano errori di snapshot del volume.

Risincronizza inversamente una relazione di replica sottoposta a failover

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

Passaggi

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

Invertire la direzione di replica dell'applicazione

Quando si inverte la direzione di replica, Trident Protect sposta l'applicazione sul backend di storage di destinazione continuando a replicare verso il backend di storage di origine. Trident Protect arresta l'applicazione di origine e replica i dati sulla destinazione prima di passare all'applicazione di destinazione.

In questa situazione, si scambiano il cluster di origine e il cluster di destinazione.

Passaggi

1. Sul cluster di origine, creare una snapshot di spegnimento:

Crea un'istantanea di spegnimento utilizzando un CR

a. Disattivare le pianificazioni della policy di protezione per l'applicazione di origine.

b. Crea un file ShutdownSnapshot CR:

i. Creare il file custom resource (CR) e assegnargli un nome (ad esempio `trident-protect-shutdownsnapshot.yaml`).

ii. Configura i seguenti attributi:

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

Esempio YAML:

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

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

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

Crea una Snapshot di spegnimento utilizzando la CLI

a. Crea la Snapshot di spegnimento, sostituendo i valori tra parentesi con le informazioni del tuo ambiente. Ad esempio:

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

2. Nel cluster di origine, dopo il completamento della Snapshot di spegnimento, ottenere lo stato della Snapshot di spegnimento:

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

3. Sul cluster di origine, trovare il valore di **shutdownsnapshot.status.appArchivePath** usando il seguente comando e registrare l'ultima parte del percorso del file (chiamata anche basename; sarà tutto ciò che si trova dopo l'ultima barra):

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

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

> Nel passo 2 della procedura di fail over, includere il `spec.promotedSnapshot` campo nel file AppMirrorRelationship CR e impostare il suo valore sul basename registrato nel passo 3 sopra.

5. Eseguire i passaggi di risincronizzazione inversa in Risincronizza inversamente una relazione di replica sottoposta a failover.
6. Abilitare le pianificazioni di protezione sul nuovo cluster di origine.

Risultato

Le seguenti azioni si verificano a causa della replica inversa:

- Viene scattata una Snapshot delle risorse Kubernetes dell'app di origine.
- I pod dell'app originale vengono interrotti in modo graduale eliminando le risorse Kubernetes dell'app (lasciando PVC e PV al loro posto).
- Dopo lo spegnimento dei pod, vengono acquisite e replicate le Snapshot dei volumi dell'applicazione.
- Le relazioni SnapMirror vengono interrotte, rendendo i volumi di destinazione pronti per la lettura/scrittura.
- Le risorse Kubernetes dell'app vengono ripristinate dalla snapshot pre-arresto, utilizzando i dati del volume replicati dopo che l'app di origine originale è stata arrestata.
- La replicazione viene ristabilita in senso inverso.

Ripristina le applicazioni nel cluster di origine

Utilizzando Trident Protect, puoi ottenere il "fail back" dopo un'operazione di failover utilizzando la seguente sequenza di operazioni. In questo flusso di lavoro per ripristinare la direzione di replica originale, Trident Protect replica (risincronizza) eventuali modifiche dell'applicazione all'applicazione sorgente originale prima di invertire la direzione di replica.

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

- Inizia con uno stato di failover.
- Invertire la risincronizzazione della relazione di replica.

> Non eseguire un'operazione di risincronizzazione normale, perché in questo modo si scartano i dati scritti sul cluster di destinazione durante la procedura di fail over.

- Invertire la direzione di replica.

Passaggi

1. Eseguire i Risincronizza inversamente una relazione di replica sottoposta a failover passaggi.
2. Eseguire i Invertire la direzione di replica dell'applicazione passaggi.

Eliminare una relazione di replica

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

Passaggi

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

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

Migra le applicazioni utilizzando Trident Protect

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

> Quando si esegue la migrazione di un'applicazione, tutti gli execution hook configurati per l'applicazione vengono migrati insieme all'app. Se è presente un execution hook post-ripristino, viene eseguito automaticamente come parte dell'operazione di ripristino.

Operazioni di backup e ripristino

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

Clona nello stesso cluster

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

Passaggi

1. 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 un cluster diverso

Per clonare un'applicazione su un cluster diverso (eseguire un clone tra cluster), crea un backup sul cluster di origine e poi ripristina il backup su un cluster di destinazione. Assicurarsi che Trident Protect sia installato sul cluster di destinazione.

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

Passaggi

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

Migrare le applicazioni da una storage class a un'altra storage class

È possibile migrare le applicazioni da una classe di storage a un'altra classe di storage ripristinando un backup nella classe di storage di destinazione.

Ad esempio (escludendo i segreti dal restore CR):

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

Ripristina l'istantanea utilizzando un CR

Passaggi

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

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

   - **spec.appVaultRef**: (*Obbligatorio*) Il nome del AppVault in cui sono archiviati i contenuti dello snapshot.
   - **spec.namespaceMapping**: Il mapping dello spazio dei nomi di origine dell'operazione di ripristino allo spazio dei nomi di destinazione. Sostituisci `my-source-namespace` e `my-destination-namespace` con le informazioni del tuo ambiente.

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

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

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

  Ad esempio:

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

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

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

Ripristina l'istantanea utilizzando la CLI

Passaggi

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

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

     Ad esempio:

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

Gestire gli hook di esecuzione di Trident Protect

Un hook di esecuzione è un'azione personalizzata che puoi configurare per essere eseguita insieme a un'operazione di protezione dei dati di un'app gestita. Ad esempio, se hai un'app di database, puoi utilizzare un hook di esecuzione per mettere in pausa tutte le transazioni del database prima di uno Snapshot e riprenderle dopo che lo Snapshot è stato completato. Questo garantisce Snapshot coerenti con l'applicazione.

Tipi di hook di esecuzione

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

- Pre-Snapshot
- Post-Snapshot
- Pre-backup
- Post-backup
- Post-ripristino
- Post-failover

Ordine di esecuzione

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

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

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

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

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

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

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

Note importanti sui ganci di esecuzione personalizzati

Considera quanto segue quando pianifichi gli hook di esecuzione per le tue app.

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

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

Filtri hook di esecuzione

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

Ogni filtro che aggiungi a un hook di esecuzione utilizza un'espressione regolare per trovare corrispondenze

con i container nel tuo cluster. Quando un hook trova una corrispondenza con un container, eseguirà lo script associato su quel container. Le espressioni regolari per i filtri utilizzano la sintassi Regular Expression 2 (RE2), che non supporta la creazione di un filtro che escluda i container dall'elenco delle corrispondenze. Per informazioni sulla sintassi che Trident Protect supporta per le espressioni regolari nei filtri degli hook di esecuzione, vedere "Supporto della sintassi Regular Expression 2 (RE2)".

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

Esempi di execution hook

Visita il "Progetto NetApp Verda GitHub" per scaricare hook di esecuzione reali per app popolari come Apache Cassandra e Elasticsearch. Puoi anche vedere esempi e ottenere idee per strutturare i tuoi hook di esecuzione personalizzati.

Crea un hook di esecuzione

È possibile creare un hook di esecuzione personalizzato per un'app utilizzando Trident Protect. Per creare hook di esecuzione, è necessario disporre delle autorizzazioni di Owner, Admin o Member.

Utilizzare un CR

Passaggi

1. Crea il file custom resource (CR) e assegnagli il nome `trident-protect-hook.yaml`.
2. Configura i seguenti attributi in modo che corrispondano al tuo ambiente Trident Protect e alla configurazione del cluster:
   - **metadata.name**: (*Obbligatorio*) Il nome di questa risorsa personalizzata; scegli un nome univoco e sensato per il tuo ambiente.
   - **spec.applicationRef**: (*Obbligatorio*) Il nome Kubernetes dell'applicazione per cui eseguire l'hook di esecuzione.
   - **spec.stage**: (*Obbligatorio*) Una stringa che indica in quale fase dell'azione deve essere eseguito l'hook di esecuzione. Valori possibili:
     - Pre
     - Post
   - **spec.action**: (*Obbligatorio*) Una stringa che indica quale azione verrà intrapresa dall'hook di esecuzione, supponendo che eventuali filtri dell'hook di esecuzione specificati corrispondano. Valori possibili:
     - Snapshot
     - Backup
     - Ripristina
     - Failover
   - **spec.enabled**: (*Facoltativo*) Indica se questo hook di esecuzione è abilitato o disabilitato. Se non specificato, il valore predefinito è true.
   - **spec.hookSource**: (*Obbligatorio*) Una stringa contenente lo script hook codificato in base64.
   - **spec.timeout**: (*Facoltativo*) Un numero che definisce per quanti minuti è consentita l'esecuzione dell'hook di esecuzione. Il valore minimo è 1 minuto e il valore predefinito è 25 minuti se non specificato.
   - **spec.arguments**: (*Facoltativo*) Un elenco YAML di argomenti che puoi specificare per l'execution hook.
   - **spec.matchingCriteria**: (*Facoltativo*) Un elenco facoltativo di coppie chiave-valore di criteri, ciascuna delle quali costituisce un filtro di hook di esecuzione. È possibile aggiungere fino a 10 filtri per hook di esecuzione.
   - **spec.matchingCriteria.type**: (*Facoltativo*) Una stringa che identifica il tipo di filtro dell'hook di esecuzione. Possibili valori:
     - ContainerImage
     - ContainerName
     - PodName
     - PodLabel
     - NamespaceName
   - **spec.matchingCriteria.value**: (*Facoltativo*) Una stringa o espressione regolare che identifica il valore del filtro dell'hook di esecuzione.

     Esempio YAML:

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

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

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

Usa la CLI

Passaggi

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

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

Esegui manualmente un hook di esecuzione

È possibile eseguire manualmente un hook di esecuzione a scopo di test o se è necessario rieseguire manualmente l'hook dopo un errore. È necessario disporre delle autorizzazioni di Owner, Admin o Member per eseguire manualmente gli hook di esecuzione.

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

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

Passaggio 1: crea un backup delle risorse

Utilizzare un CR

Passaggi

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

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

     Esempio YAML:

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

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

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

Usa la CLI

Passaggi

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

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

2. Visualizza lo stato del backup. Puoi usare questo comando di esempio ripetutamente fino al completamento dell'operazione:

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

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

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

Passaggio 2: eseguire l’hook di esecuzione

Utilizzare un CR

Passaggi

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

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

   - **spec.action**: (*Obbligatorio*) Una stringa che indica quale azione verrà intrapresa dall'hook di esecuzione, supponendo che eventuali filtri dell'hook di esecuzione specificati corrispondano. Valori possibili:
     - Snapshot
     - Backup
     - Ripristina
     - Failover
   - **spec.stage**: (*Obbligatorio*) Una stringa che indica in quale fase dell'azione deve essere eseguito l'hook di esecuzione. Questa esecuzione dell'hook non eseguirà hook in nessun'altra fase. Valori possibili:
     - Pre
     - Post

     Esempio YAML:

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

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

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

Usa la CLI

Passaggi

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

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

2. Controlla lo stato dell'esecuzione del hook. Puoi eseguire questo comando ripetutamente fino al completamento dell'operazione:

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

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

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

Informazioni sul copyright

Informazioni sul marchio commerciale