Managen und sichern Sie Applikationen

Trident

NetApp
January 14, 2026

Inhalt

Managen und sichern Sie Applikationen

Verwenden Sie Trident Protect AppVault-Objekte, um Buckets zu verwalten.

Die Bucket Custom Resource (CR) für Trident Protect wird als AppVault bezeichnet. AppVault-Objekte sind die deklarative Kubernetes-Workflow-Darstellung eines Speicher-Buckets. Ein AppVault CR enthält die Konfigurationen, die für die Verwendung eines Buckets in Schutzvorgängen wie Backups, Snapshots, Wiederherstellungsvorgängen und SnapMirror Replikation erforderlich sind. Nur Administratoren können AppVaults erstellen.

Sie müssen einen AppVault CR entweder manuell oder über die Befehlszeile erstellen, wenn Sie Datensicherungsvorgänge an einer Anwendung durchführen, und der AppVault CR muss sich auf dem Cluster befinden, auf dem Trident Protect installiert ist. Der AppVault CR ist spezifisch für Ihre Umgebung; Sie können die Beispiele auf dieser Seite als Leitfaden beim Erstellen von AppVault CRs verwenden.

Konfigurieren Sie die AppVault-Authentifizierung und Passwörter

Bevor Sie ein AppVault CR-System erstellen, müssen Sie sicherstellen, dass sich AppVault und der von Ihnen ausgewählte Data Mover beim Anbieter und den zugehörigen Ressourcen authentifizieren können.

Passwörter für Data Mover Repository

Wenn Sie AppVault-Objekte mithilfe von CRs oder dem Trident Protect CLI-Plugin erstellen, können Sie Trident Protect optional anweisen, ein Kubernetes-Secret zu verwenden, das benutzerdefinierte Passwörter für die Verschlüsselung des Restic- und Kopia-Repositorys enthält. Wenn Sie kein Geheimnis angeben, verwendet Trident Protect ein Standardpasswort.

- Wenn Sie AppVault CRs manuell erstellen, verwenden Sie das Feld **spec.dataMoverPasswordSecretRef**, um das Geheimnis anzugeben.
- Verwenden Sie beim Erstellen von AppVault-Objekten mit der Trident Protect CLI die `--data-mover-password-secret-ref` Argument zur Angabe des Geheimnisses.

Erstellen Sie einen Kennwortschlüssel für das Data Mover Repository

Nutzen Sie die folgenden Beispiele, um das Passwortgeheimnis zu erstellen. Wenn Sie AppVault-Objekte erstellen, können Sie Trident Protect anweisen, dieses Geheimnis zur Authentifizierung beim Datenübertragungsrepository zu verwenden.

Je nachdem, welchen Data Mover Sie verwenden, müssen Sie nur das entsprechende Passwort für diesen Data Mover angeben. Wenn Sie beispielsweise Restic verwenden und Kopia in Zukunft nicht verwenden möchten, können Sie beim Erstellen des Geheimnisses nur das Restic-Kennwort eingeben.

CR verwenden

```
---
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
```

Verwenden Sie die 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
```

S3-kompatible Speicher-IAM-Berechtigungen

Wenn Sie auf S3-kompatiblen Speicher wie Amazon S3, Generic S3, zugreifen "StorageGRID S3" , oder "ONTAP S3" Bei der Verwendung von Trident Protect müssen Sie sicherstellen, dass die von Ihnen angegebenen Benutzerdaten über die erforderlichen Berechtigungen für den Zugriff auf den Bucket verfügen. Nachfolgend ein Beispiel für eine Richtlinie, die die minimal erforderlichen Berechtigungen für den Zugriff mit Trident Protect gewährt. Sie können diese Richtlinie auf den Benutzer anwenden, der S3-kompatible Bucket-Richtlinien verwaltet.

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

Weitere Informationen zu Amazon S3-Richtlinien finden Sie in den Beispielen im "Amazon S3-Dokumentation" .

Beispiele für die Schlüsselgeneration von AppVault für Cloud-Provider

Beim Definieren eines AppVault CR müssen Sie Anmeldeinformationen eingeben, um auf die vom Anbieter gehosteten Ressourcen zugreifen zu können. Die Art und Weise, wie Sie die Schlüssel für die Anmeldeinformationen generieren, hängt vom Anbieter ab. Im Folgenden finden Sie Beispiele für die Generierung von Kommandozeilen-Schlüsseln für mehrere Anbieter. In den folgenden Beispielen können Sie Schlüssel für die Anmeldedaten der einzelnen Cloud-Provider erstellen.

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
```

Allgemein S3

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

ONTAP S3

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

StorageGRID S3

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

Beispiele für die Erstellung von AppVault

Im Folgenden finden Sie Beispiele für AppVault-Definitionen für jeden Anbieter.

AppVault CR-Beispiele

Sie können die folgenden CR-Beispiele verwenden, um AppVault-Objekte für jeden Cloud-Provider zu erstellen.

- Sie können optional einen Kubernetes-Schlüssel angeben, der benutzerdefinierte Passwörter für die Restic- und Kopia-Repository-Verschlüsselung enthält. Weitere Informationen finden Sie unter Passwörter für Data Mover Repository .
- Für Amazon S3 (AWS) AppVault-Objekte können Sie optional ein SessionToken angeben, was nützlich ist, wenn Sie Single Sign-On (SSO) für die Authentifizierung verwenden. Dieses Token wird erstellt, wenn Sie Schlüssel für den Provider in generierenBeispiele für die Schlüsselgeneration von AppVault für Cloud-Provider.
- Für S3 AppVault-Objekte können Sie optional eine Proxy-URL für ausgehenden S3-Datenverkehr über den Schlüssel angeben `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
```

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
```

Allgemein S3

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

ONTAP S3

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

StorageGRID S3

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

Beispiele zur AppVault-Erstellung mit der Trident Protect CLI

Sie können die folgenden CLI-Befehlsbeispiele verwenden, um AppVault CRS für jeden Anbieter zu erstellen.

- Sie können optional einen Kubernetes-Schlüssel angeben, der benutzerdefinierte Passwörter für die Restic- und Kopia-Repository-Verschlüsselung enthält. Weitere Informationen finden Sie unter Passwörter für Data Mover Repository .
- Für S3-AppVault-Objekte können Sie optional mithilfe des Arguments eine Proxy-URL für ausgehenden S3-Datenverkehr angeben `--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
```

Allgemein S3

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

ONTAP S3

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

StorageGRID S3

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

Informationen zu AppVault anzeigen

Mit dem Trident Protect CLI-Plugin können Sie Informationen über AppVault-Objekte anzeigen, die Sie auf dem Cluster erstellt haben.

Schritte

1. Inhalt eines AppVault-Objekts anzeigen:

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

   Beispielausgabe:

```
+-------------+-------+----------+-----------------------------
+---------------------------+
|   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. Um den AppVaultPath für jede Ressource anzuzeigen, verwenden Sie optional das Flag `--show-paths`.

   Der Clustername in der ersten Spalte der Tabelle ist nur verfügbar, wenn bei der Trident Protect Helm-Installation ein Clustername angegeben wurde. Zum Beispiel: `--set clusterName=production1` .

Entfernen Sie einen AppVault

Sie können ein AppVault-Objekt jederzeit entfernen.

> Entfernen Sie den Schlüssel im AppVault CR nicht `finalizers`, bevor Sie das AppVault-Objekt löschen. Wenn Sie dies tun, kann dies zu Restdaten im AppVault-Bucket und verwaisten Ressourcen im Cluster führen.

Bevor Sie beginnen

Stellen Sie sicher, dass Sie alle Snapshot- und Backup-CRS gelöscht haben, die vom AppVault verwendet werden, den Sie löschen möchten.

Entfernen Sie einen AppVault mithilfe der Kubernetes-CLI

1. Entfernen Sie das AppVault-Objekt und ersetzen Sie `appvault-name` es durch den Namen des zu entfernenden AppVault-Objekts:

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

Entfernen Sie einen AppVault mithilfe der Trident Protect CLI.

1. Entfernen Sie das AppVault-Objekt und ersetzen Sie `appvault-name` es durch den Namen des zu entfernenden AppVault-Objekts:

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

Definieren Sie eine Anwendung für die Verwaltung mit Trident Protect.

Sie können eine Anwendung, die Sie mit Trident Protect verwalten möchten, definieren, indem Sie eine Anwendungs-CR und eine zugehörige AppVault-CR erstellen.

Erstellen Sie ein AppVault CR

Sie müssen einen AppVault CR erstellen, der bei der Durchführung von Datenschutzoperationen an der Anwendung verwendet wird, und der AppVault CR muss sich auf dem Cluster befinden, auf dem Trident Protect installiert ist. Die AppVault-CR ist spezifisch für Ihre Umgebung; Beispiele für AppVault-CRs finden Sie unter"Benutzerdefinierte Ressourcen von AppVault."

Definieren Sie eine Anwendung

Sie müssen jede Anwendung definieren, die Sie mit Trident Protect verwalten möchten. Sie können eine Anwendung zur Verwaltung definieren, indem Sie entweder manuell eine Anwendungs-CR erstellen oder die Trident Protect CLI verwenden.

Fügen Sie eine Anwendung mit einem CR hinzu

Schritte

1. Erstellen Sie die CR-Datei der Zielanwendung:
   a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (z. B. `maria-app.yaml`).
   b. Konfigurieren Sie die folgenden Attribute:
      - **metadata.name**: (*required*) der Name der benutzerdefinierten Ressource der Anwendung. Beachten Sie den von Ihnen ausgewählten Namen, da sich andere CR-Dateien, die für Schutzvorgänge benötigt werden, auf diesen Wert beziehen.
      - **spec.includedNamespaces**: (*required*) Verwenden Sie Namespace und Label Selector, um die Namespaces und Ressourcen anzugeben, die die Anwendung verwendet. Der Application Namespace muss Teil dieser Liste sein. Die Label Selector ist optional und kann verwendet werden, um Ressourcen innerhalb jedes angegebenen Namespace zu filtern.
      - **spec.includedClusterScopedResources**: (*Optional*) Verwenden Sie dieses Attribut, um Clusterressourcen anzugeben, die in die Anwendungsdefinition aufgenommen werden sollen. Mit diesem Attribut können Sie diese Ressourcen anhand ihrer Gruppe, Version, Art und Labels auswählen.
        - **GroupVersionRind**: (*required*) gibt die API-Gruppe, die Version und die Art der Clusterressource an.
        - **LabelSelector**: (*Optional*) filtert die Cluster-Ressourcen basierend auf ihren Bezeichnungen.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Optional*) Diese Annotation ist nur für Anwendungen anwendbar, die von virtuellen Maschinen definiert werden, wie z. B. in KubeVirt-Umgebungen, wo Dateisystem-Freezes vor Snapshots auftreten. Legen Sie fest, ob diese Anwendung während einer Snapshot-Erstellung auf das Dateisystem schreiben darf. Wenn diese Option auf „true“ gesetzt ist, ignoriert die Anwendung die globale Einstellung und kann während eines Snapshots auf das Dateisystem schreiben. Wenn der Wert auf „false“ gesetzt ist, ignoriert die Anwendung die globale Einstellung und das Dateisystem wird während eines Snapshots eingefroren. Wird dies angegeben, die Anwendung aber keine virtuellen Maschinen in der Anwendungsdefinition aufweist, wird die Annotation ignoriert. Sofern nichts anderes angegeben ist, gilt die folgende Vorgehensweise:"Globale Trident Protect-Einfrierungseinstellung" .

Wenn Sie diese Anmerkung anwenden müssen, nachdem eine Anwendung bereits erstellt wurde, können Sie den folgenden Befehl verwenden:

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

+
Beispiel 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. Nachdem Sie die CR-Anwendung erstellt haben, die Ihrer Umgebung entspricht, wenden Sie den CR an. Beispiel:

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

Schritte

1. Erstellen und wenden Sie die Anwendungsdefinition anhand eines der folgenden Beispiele an. Ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung. Sie können Namespaces und Ressourcen in die Anwendungsdefinition mit kommagetrennten Listen mit den in den Beispielen gezeigten Argumenten aufnehmen.

   Optional können Sie beim Erstellen einer App eine Annotation verwenden, um anzugeben, ob die Anwendung während eines Snapshots auf das Dateisystem schreiben darf. Dies gilt nur für Anwendungen, die von virtuellen Maschinen definiert werden, wie beispielsweise in KubeVirt-

Umgebungen, wo Dateisystem-Freezes vor Snapshots auftreten. Wenn Sie die Annotation auf `true` Die Anwendung ignoriert die globale Einstellung und kann während eines Snapshots in das Dateisystem schreiben. Wenn Sie es einstellen auf `false` Die Anwendung ignoriert die globale Einstellung und das Dateisystem wird während eines Snapshots eingefroren. Wenn Sie die Annotation verwenden, die Anwendung aber keine virtuellen Maschinen in der Anwendungsdefinition aufweist, wird die Annotation ignoriert. Wenn Sie die Annotation nicht verwenden, folgt die Anwendung dem Standard."Globale Trident Protect-Einfrierungseinstellung" .

Um die Anmerkung anzugeben, wenn Sie die CLI zum Erstellen einer Anwendung verwenden, können Sie das Flag verwenden `--annotation`.

- Erstellen Sie die Anwendung, und verwenden Sie die globale Einstellung für das Verhalten des Dateisystemfixieren:

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

- Erstellen Sie die Anwendung, und konfigurieren Sie die lokale Anwendungseinstellung für das Dateisystem-Standverhalten:

```
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">
```

Schützen Sie Ihre Anwendungen mit Trident Protect.

Sie können alle von Trident Protect verwalteten Apps schützen, indem Sie mithilfe einer automatisierten Schutzrichtlinie oder ad hoc Snapshots und Backups erstellen.

Sie können Trident Protect so konfigurieren, dass Dateisysteme während Datensicherungsvorgängen eingefroren und wieder freigegeben werden. "Erfahren Sie mehr über die Konfiguration des Einfrierens von Dateisystemen mit Trident Protect.".

Erstellen Sie einen On-Demand Snapshot

Sie können jederzeit einen On-Demand-Snapshot erstellen.

Im Umfang des Clusters enthaltene Ressourcen werden in einem Backup, einem Snapshot oder Klon eingeschlossen, wenn sie in der Anwendungsdefinition explizit referenziert werden oder wenn sie Verweise auf einen der Anwendungsnamepspaces haben.

Erstellen Sie mithilfe eines CR-Systems einen Snapshot

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-snapshot-cr.yaml`.
2. Konfigurieren Sie in der erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*required*) der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **Spec.applicationRef**: Der Kubernetes-Name der zu Snapshot enden Anwendung.
   - **Spec.appVaultRef**: (*required*) der Name des AppVault, in dem der Snapshot-Inhalt (Metadaten) gespeichert werden soll.
   - **Spec.reclaimPolicy**: (*Optional*) definiert, was mit dem AppArchiv eines Snapshots geschieht, wenn der Snapshot CR gelöscht wird. Das bedeutet, dass der Snapshot auch dann gelöscht wird, wenn er auf gesetzt `Retain` ist. Gültige Optionen:
     - `Retain` (Standard)
     - `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. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-snapshot-cr.yaml` haben, wenden Sie den CR an:

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

Erstellen Sie einen Snapshot mithilfe der CLI

Schritte

1. Erstellen Sie den Snapshot, und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung. Beispiel:

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

Erstellen Sie ein On-Demand-Backup

Sie können eine App jederzeit sichern.

> Im Umfang des Clusters enthaltene Ressourcen werden in einem Backup, einem Snapshot oder Klon eingeschlossen, wenn sie in der Anwendungsdefinition explizit referenziert werden oder wenn sie Verweise auf einen der Anwendungsnamepspaces haben.

Bevor Sie beginnen

Vergewissern Sie sich, dass der Ablauf des AWS-Sitzungstokens für alle langwierigen s3-Backup-Vorgänge ausreichend ist. Wenn das Token während des Backup-Vorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Überprüfen des Ablaufes des aktuellen Sitzungstokens finden Sie im "AWS API-Dokumentation".
- Weitere Informationen zu Zugangsdaten für AWS Ressourcen finden Sie in der "AWS IAM-Dokumentation".

Erstellen Sie ein Backup mit einem CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-backup-cr.yaml`.
2. Konfigurieren Sie in der erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*required*) der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **Spec.applicationRef**: (*required*) der Kubernetes-Name der zu Back-up-Applikation.
   - **Spec.appVaultRef**: (*required*) der Name des AppVault, in dem der Backup-Inhalt gespeichert werden soll.
   - **Spec.DataMover**: (*Optional*) Eine Zeichenfolge, die angibt, welches Backup-Tool für den Backup-Vorgang verwendet werden soll. Mögliche Werte (Groß-/Kleinschreibung beachten):
     - `Restic`
     - `Kopia` (Standard)
   - **Spec.reclaimPolicy**: (*Optional*) definiert, was mit einem Backup geschieht, wenn es von seinem Anspruch freigegeben wird. Mögliche Werte:
     - `Delete`
     - `Retain` (Standard)
   - **Spec.snapshotRef**: (*Optional*): Name des als Quelle des Backups zu verwendenden Snapshot. Falls nicht angegeben, wird ein temporärer Snapshot erstellt und gesichert.

     Beispiel YAML:

```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. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-backup-cr.yaml` haben, wenden Sie den CR an:

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

Erstellen Sie mithilfe der CLI ein Backup

Schritte

1. Erstellen Sie das Backup, und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung. Beispiel:

```
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>
```

Erstellen Sie einen Zeitplan für die Datensicherung

Eine Sicherungsrichtlinie sichert eine Applikation, indem Snapshots, Backups oder beides nach einem definierten Zeitplan erstellt werden. Sie können Snapshots und Backups stündlich, täglich, wöchentlich und monatlich erstellen. Außerdem können Sie die Anzahl der beizubehaltenden Kopien festlegen.

Im Umfang des Clusters enthaltene Ressourcen werden in einem Backup, einem Snapshot oder Klon eingeschlossen, wenn sie in der Anwendungsdefinition explizit referenziert werden oder wenn sie Verweise auf einen der Anwendungsnamepspaces haben.

Bevor Sie beginnen

Vergewissern Sie sich, dass der Ablauf des AWS-Sitzungstokens für alle langwierigen s3-Backup-Vorgänge ausreichend ist. Wenn das Token während des Backup-Vorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Überprüfen des Ablaufes des aktuellen Sitzungstokens finden Sie im "AWS API-Dokumentation".
- Weitere Informationen zu Zugangsdaten für AWS Ressourcen finden Sie in der "AWS IAM-Dokumentation".

Erstellen Sie einen Zeitplan mit einem CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-schedule-cr.yaml`.
2. Konfigurieren Sie in der erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*required*) der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **Spec.DataMover**: (*Optional*) Eine Zeichenfolge, die angibt, welches Backup-Tool für den Backup-Vorgang verwendet werden soll. Mögliche Werte (Groß-/Kleinschreibung beachten):
     - `Restic`
     - `Kopia` (Standard)
   - **Spec.applicationRef**: Der Kubernetes-Name der zu Back-up Applikation.
   - **Spec.appVaultRef**: (*required*) der Name des AppVault, in dem der Backup-Inhalt gespeichert werden soll.
   - **Spec.backupRetention**: Die Anzahl der zu behaltenden Backups. Null bedeutet, dass keine Backups erstellt werden sollen.
   - **Spec.snapshotRetention**: Die Anzahl der zu behaltenden Snapshots. Null bedeutet, dass keine Snapshots erstellt werden sollen.
   - **spec.granularity**: die Häufigkeit, mit der der Zeitplan ausgeführt werden soll. Mögliche Werte, zusammen mit den erforderlichen zugeordneten Feldern:
     - `Hourly`(erfordert die Angabe `spec.minute`)
     - `Daily`(erfordert die Angabe `spec.minute` Und `spec.hour`)
     - `Weekly`(erfordert die Angabe `spec.minute, spec.hour` , Und `spec.dayOfWeek`)
     - `Monthly`(erfordert die Angabe `spec.minute, spec.hour` , Und `spec.dayOfMonth`)
     - `Custom`
   - **spec.dayOfMonth**: (*Optional*) Der Tag des Monats (1 – 31), an dem der Zeitplan ausgeführt werden soll. Dieses Feld ist erforderlich, wenn die Granularität auf `Monthly` .
   - **spec.dayOfWeek**: (*Optional*) Der Wochentag (0 - 7), an dem der Zeitplan ausgeführt werden soll. Werte von 0 oder 7 zeigen Sonntag an. Dieses Feld ist erforderlich, wenn die Granularität auf `Weekly` .
   - **spec.hour**: (*Optional*) Die Stunde des Tages (0 - 23), zu der der Zeitplan ausgeführt werden soll. Dieses Feld ist erforderlich, wenn die Granularität auf `Daily` , `Weekly` , oder `Monthly` .
   - **spec.minute**: (*Optional*) Die Minute der Stunde (0 – 59), zu der der Zeitplan ausgeführt werden soll. Dieses Feld ist erforderlich, wenn die Granularität auf `Hourly` , `Daily` , `Weekly` , oder `Monthly` .

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: Monthly
  dayOfMonth: "1"
  dayOfWeek: "0"
  hour: "0"
  minute: "0"
```

3. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-schedule-cr.yaml` haben, wenden Sie den CR an:

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

Erstellen Sie einen Zeitplan über die CLI

Schritte

1. Erstellen Sie den Schutzplan und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung. Beispiel:

> Mit können `tridentctl-protect create schedule --help` Sie detaillierte Hilfeinformationen für diesen Befehl anzeigen.

```
tridentctl-protect create schedule <my_schedule_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> --backup
-retention <how_many_backups_to_retain> --data-mover
<Kopia_or_Restic> --day-of-month <day_of_month_to_run_schedule>
--day-of-week <day_of_month_to_run_schedule> --granularity
<frequency_to_run> --hour <hour_of_day_to_run> --minute
<minute_of_hour_to_run> --recurrence-rule <recurrence> --snapshot
-retention <how_many_snapshots_to_retain> -n <application_namespace>
```

Löschen Sie einen Snapshot

Löschen Sie die geplanten oder On-Demand Snapshots, die Sie nicht mehr benötigen.

Schritte

1. Entfernen Sie den Snapshot CR, der dem Snapshot zugeordnet ist:

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

Löschen Sie ein Backup

Löschen Sie die geplanten oder On-Demand-Backups, die Sie nicht mehr benötigen.

Schritte

1. Entfernen Sie den Backup-CR, der dem Backup zugeordnet ist:

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

Überprüfen Sie den Status eines Sicherungsvorgangs

Sie können die Befehlszeile verwenden, um den Status eines laufenden, abgeschlossenen oder fehlgeschlagenen Sicherungsvorgangs zu überprüfen.

Schritte

1. Verwenden Sie den folgenden Befehl, um den Status des Sicherungsvorgangs abzurufen und Werte in Bracken durch Informationen aus Ihrer Umgebung zu ersetzen:

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

Backup und Restore für Azure-NetApp-Files (ANF)-Vorgänge

Wenn Sie Trident Protect installiert haben, können Sie die speichereffiziente Sicherungs- und Wiederherstellungsfunktionalität für Speicher-Backends aktivieren, die die Speicherklasse azure-netapp-files verwenden und vor Trident 24.06 erstellt wurden. Diese Funktionalität funktioniert mit NFSv4-Volumes und verbraucht keinen zusätzlichen Speicherplatz aus dem Kapazitätspool.

Bevor Sie beginnen

Stellen Sie Folgendes sicher:

- Sie haben Trident Protect installiert.
- Sie haben eine Anwendung in Trident Protect definiert. Diese Anwendung bietet nur eingeschränkten Schutz, bis Sie dieses Verfahren abgeschlossen haben.
- Sie haben `azure-netapp-files` als Standard-Storage-Klasse für Ihr Storage-Back-End ausgewählt.

Erweitern Sie für Konfigurationsschritte

1. Gehen Sie in Trident folgendermaßen vor, wenn das ANF-Volume vor dem Upgrade auf Trident 24.10 erstellt wurde:

   a. Aktivieren Sie das Snapshot-Verzeichnis für jedes PV, das auf Azure-NetApp-Dateien basiert und der Anwendung zugeordnet ist:

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

   b. Vergewissern Sie sich, dass das Snapshot-Verzeichnis für jedes zugeordnete PV aktiviert wurde:

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

   Antwort:

   ```
   snapshotDirectory: "true"
   ```

   +
   Wenn das Snapshot-Verzeichnis nicht aktiviert ist, wählt Trident Protect die reguläre Backup-Funktionalität, die während des Backup-Vorgangs vorübergehend Speicherplatz im Kapazitätspool belegt. Stellen Sie in diesem Fall sicher, dass im Kapazitätspool ausreichend Speicherplatz vorhanden ist, um ein temporäres Volume in der Größe des zu sichernden Volumes zu erstellen.

Ergebnis

Die Anwendung ist für die Datensicherung und -wiederherstellung mit Trident Protect bereit. Jede PVC kann auch von anderen Anwendungen für Backups und Wiederherstellungen verwendet werden.

Anwendungen mit Trident Protect wiederherstellen

Mit Trident Protect können Sie Ihre Anwendung aus einem Snapshot oder einer Sicherung wiederherstellen. Die Wiederherstellung aus einem vorhandenen Snapshot ist schneller, wenn die Anwendung im selben Cluster wiederhergestellt wird.

Wenn Sie eine Anwendung wiederherstellen, werden alle für die Anwendung konfigurierten Ausführungshaken mit der App wiederhergestellt. Wenn ein Hook für die Ausführung nach der Wiederherstellung vorhanden ist, wird er automatisch als Teil des Wiederherstellungsvorgangs ausgeführt.

Namespace-Annotationen und -Labels bei Restore- und Failover-Vorgängen

Während von Restore- und Failover-Vorgängen werden Labels und Annotationen im Ziel-Namespace so angefertigt, dass sie den Labels und Annotationen im Quell-Namespace entsprechen. Beschriftungen oder Anmerkungen aus dem Quell-Namespace, die nicht im Ziel-Namespace vorhanden sind, werden hinzugefügt.

Alle bereits vorhandenen Beschriftungen oder Anmerkungen werden überschrieben, um dem Wert aus dem Quell-Namespace zu entsprechen. Beschriftungen oder Anmerkungen, die nur im Ziel-Namespace vorhanden sind, bleiben unverändert.

> Wenn Sie Red hat OpenShift verwenden, ist es wichtig, die wichtige Rolle von Namespace-Annotationen in OpenShift-Umgebungen zu beachten. Namespace-Anmerkungen stellen sicher, dass wiederhergestellte Pods den durch OpenShift Security Context Constraints (SCCs) definierten Berechtigungen und Sicherheitskonfigurationen entsprechen und ohne Berechtigungsprobleme auf Volumes zugreifen können. Weitere Informationen finden Sie im "Dokumentation zu den Einschränkungen für den Sicherheitskontext von OpenShift".

Sie können verhindern, dass bestimmte Annotationen im Ziel-Namespace überschrieben werden, indem Sie die Kubernetes-Umgebungsvariable vor der Wiederherstellung oder dem Failover einstellen `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`. Beispiel:

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

Wenn Sie die Quellanwendung mit Helm installiert haben `--create-namespace` Flagge, besondere Behandlung wird der `name` Legende mit Beschriftung. Während des Wiederherstellungs- oder Failover-Prozesses kopiert Trident Protect diese Bezeichnung in den Ziel-Namespace, aktualisiert aber den Wert auf den Wert des Ziel-Namespace, wenn der Wert aus der Quelle mit dem Quell-Namespace übereinstimmt. Stimmt dieser Wert nicht mit dem Quell-Namespace überein, wird er unverändert in den Ziel-Namespace kopiert.

Beispiel

Im folgenden Beispiel wird ein Quell- und Ziel-Namespace präsentiert, der jeweils unterschiedliche Annotationen und Labels enthält. Sie können den Status des Ziel-Namespace vor und nach dem Vorgang anzeigen und sehen, wie die Annotationen und Labels im Ziel-Namespace kombiniert oder überschrieben werden.

Vor der Wiederherstellung oder dem Failover

Die folgende Tabelle zeigt den Status der Beispiel-Namespaces für Quelle und Ziel vor dem Restore- oder Failover-Vorgang:

Namespace	Anmerkungen	Etiketten
Namespace ns-1 (Quelle)	• Annotation.one/key: „Aktualisiertwert“ • Anmerkung.zwei/Taste: "Wahr"	• Umgebung = Produktion • Compliance=hipaa • Name=ns-1
Namespace ns-2 (Ziel)	• Anmerkung.One/key: „True“ • Anmerkung.drei/Taste: "False"	• Rolle=Datenbank

Nach dem Wiederherstellungsvorgang

In der folgenden Tabelle wird der Status des Beispiel-Ziel-Namespace nach der Wiederherstellung oder dem Failover-Vorgang dargestellt. Einige Schlüssel wurden hinzugefügt, einige wurden überschrieben und das Label wurde aktualisiert, `name` um dem Ziel-Namespace zu entsprechen:

Namespace	Anmerkungen	Etiketten
Namespace ns-2 (Ziel)	• Annotation.one/key: „Aktualisiertwert“ • Anmerkung.zwei/Taste: "Wahr" • Anmerkung.drei/Taste: "False"	• Name=ns-2 • Compliance=hipaa • Umgebung = Produktion • Rolle=Datenbank

Wiederherstellung aus einem Backup in einem anderen Namespace

Wenn Sie eine Sicherung mithilfe eines BackupRestore CR in einen anderen Namespace wiederherstellen, stellt Trident Protect die Anwendung in einem neuen Namespace wieder her und erstellt einen Anwendungs-CR für die wiederhergestellte Anwendung. Um die wiederhergestellte Anwendung zu schützen, erstellen Sie bei Bedarf Backups oder Snapshots oder legen Sie einen Schutzzeitplan fest.

> Wenn Sie ein Backup in einem anderen Namespace mit vorhandenen Ressourcen wiederherstellen, werden keine Ressourcen verändert, die Namen mit denen im Backup teilen. Um alle Ressourcen im Backup wiederherzustellen, löschen Sie entweder den Ziel-Namespace und erstellen Sie ihn neu, oder stellen Sie das Backup in einem neuen Namespace wieder her.

Bevor Sie beginnen

Vergewissern Sie sich, dass der Ablauf des AWS-Sitzungstokens für alle langwierigen s3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Überprüfen des Ablaufes des aktuellen Sitzungstokens finden Sie im "AWS API-Dokumentation".
- Weitere Informationen zu Zugangsdaten für AWS Ressourcen finden Sie in der "AWS IAM-Dokumentation".

> Wenn Sie Backups mit Kopia als Datenmigrationsprogramm wiederherstellen, können Sie optional Anmerkungen in der CR oder über die CLI angeben, um das Verhalten des von Kopia verwendeten temporären Speichers zu steuern. Siehe die "Kopia-Dokumentation" Weitere Informationen zu den konfigurierbaren Optionen finden Sie hier. Verwenden Sie die `tridentctl-protect create --help` Befehl für weitere Informationen zum Festlegen von Annotationen mit der Trident Protect CLI.

CR verwenden

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-backup-restore-cr.yaml`.
2. Konfigurieren Sie in der erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*required*) der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **Spec.appArchivePath**: Der Pfad innerhalb von AppVault, in dem die Backup-Inhalte gespeichert werden. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

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

   - **Spec.appVaultRef**: (*required*) der Name des AppVault, in dem der Backup-Inhalt gespeichert ist.
   - **spec.namespaceMapping**: die Zuordnung des Quell-Namespace des Wiederherstellungsvorgangs zum Ziel-Namespace. Ersetzen `my-source-namespace` Sie und `my-destination-namespace` mit Informationen aus Ihrer Umgebung.
   - **Spec.storageClassMapping**: Das Mapping der Quellspeicherklasse des Wiederherstellungsvorgangs an die Zielspeicherklasse. Ersetzen `destinationStorageClass` Sie und `sourceStorageClass` mit Informationen aus Ihrer Umgebung.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: BackupRestore
     metadata:
       name: my-cr-name
       namespace: my-destination-namespace
       annotations: # Optional annotations for Kopia data mover
         protect.trident.netapp.io/kopia-content-cache-size-limit-mb:
     "1000"
     spec:
       appArchivePath: my-backup-path
       appVaultRef: appvault-name
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
       storageClassMapping:
         destination: "${destinationStorageClass}"
         source: "${sourceStorageClass}"
     ```

3. (*Optional*) Wenn Sie nur bestimmte Ressourcen der wiederherzustellenden Anwendung auswählen müssen, fügen Sie eine Filterung hinzu, die Ressourcen mit bestimmten Bezeichnungen enthält oder ausschließt:

Trident Protect wählt einige Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn Sie beispielsweise eine persistente Volume-Claim-Ressource auswählen und diese einen zugehörigen Pod hat, stellt Trident Protect auch den zugehörigen Pod wieder her.

- **ResourceFilter.resourceSelectionCriteria**: (Erforderlich für die Filterung) Verwenden `Include` oder `Exclude` um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die einzuschließenden oder auszuschließenden Ressourcen zu definieren:
  - **RefindeFilter.refindeMatchers**: Eine Reihe von refindeMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, stimmen sie als OP-Operation überein, und die Felder innerhalb jedes Elements (Gruppe, Typ, Version) stimmen mit einer UND-Operation überein.
    - **ResourceMatchers[].Group**: (*Optional*) Gruppe der zu filternden Ressource.
    - **ResourceMatchers[].Kind**: (*Optional*) Art der zu filternden Ressource.
    - **ResourceMatchers[].Version**: (*Optional*) Version der zu filternden Ressource.
    - **ResourceMatchers[].Namen**: (*Optional*) Namen im Kubernetes metadata.name-Feld der zu filternden Ressource.
    - **ResourceMatchers[].Namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der zu filternden Ressource.
    - **ResourceMatchers[].labelSelectors**: (*Optional*) Label selector string im Feld Kubernetes metadata.name der Ressource, wie im definiert "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

      Beispiel:

```
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. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-backup-restore-cr.yaml` haben, wenden Sie den CR an:

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

Verwenden Sie die CLI

Schritte

1. Stellen Sie das Backup in einem anderen Namespace wieder her und ersetzen Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung. Das `namespace-mapping` Argument verwendet durch Doppelpunkte getrennte Namespaces, um Quellnamepaces im Format den richtigen Zielnamepaces zuzuordnen `source1:dest1,source2:dest2`. Beispiel:

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

Wiederherstellung von einem Backup in den ursprünglichen Namespace

Sie können ein Backup im ursprünglichen Namespace jederzeit wiederherstellen.

Bevor Sie beginnen

Vergewissern Sie sich, dass der Ablauf des AWS-Sitzungstokens für alle langwierigen s3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Überprüfen des Ablaufes des aktuellen Sitzungstokens finden Sie im "AWS API-Dokumentation".
- Weitere Informationen zu Zugangsdaten für AWS Ressourcen finden Sie in der "AWS IAM-Dokumentation".

> ⓘ Wenn Sie Backups mit Kopia als Datenmigrationsprogramm wiederherstellen, können Sie optional Anmerkungen in der CR oder über die CLI angeben, um das Verhalten des von Kopia verwendeten temporären Speichers zu steuern. Siehe die "Kopia-Dokumentation" Weitere Informationen zu den konfigurierbaren Optionen finden Sie hier. Verwenden Sie die `tridentctl-protect create --help` Befehl für weitere Informationen zum Festlegen von Annotationen mit der Trident Protect CLI.

CR verwenden

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-backup-ipr-cr.yaml`.
2. Konfigurieren Sie in der erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*required*) der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **Spec.appArchivePath**: Der Pfad innerhalb von AppVault, in dem die Backup-Inhalte gespeichert werden. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

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

   - **Spec.appVaultRef**: (*required*) der Name des AppVault, in dem der Backup-Inhalt gespeichert ist.

     Beispiel:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
  annotations: # Optional annotations for Kopia data mover
    protect.trident.netapp.io/kopia-content-cache-size-limit-mb:
"1000"
spec:
  appArchivePath: my-backup-path
  appVaultRef: appvault-name
```

3. (*Optional*) Wenn Sie nur bestimmte Ressourcen der wiederherzustellenden Anwendung auswählen müssen, fügen Sie eine Filterung hinzu, die Ressourcen mit bestimmten Bezeichnungen enthält oder ausschließt:

   Trident Protect wählt einige Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn Sie beispielsweise eine persistente Volume-Claim-Ressource auswählen und diese einen zugehörigen Pod hat, stellt Trident Protect auch den zugehörigen Pod wieder her.

   - **ResourceFilter.resourceSelectionCriteria**: (Erforderlich für die Filterung) Verwenden `Include` oder `Exclude` um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die einzuschließenden oder auszuschließenden Ressourcen zu definieren:
     - **RefindeFilter.refindeMatchers**: Eine Reihe von refindeMatcher-Objekten. Wenn Sie mehrere

Elemente in diesem Array definieren, stimmen sie als OP-Operation überein, und die Felder innerhalb jedes Elements (Gruppe, Typ, Version) stimmen mit einer UND-Operation überein.

- **ResourceMatchers[].Group**: (*Optional*) Gruppe der zu filternden Ressource.
- **ResourceMatchers[].Kind**: (*Optional*) Art der zu filternden Ressource.
- **ResourceMatchers[].Version**: (*Optional*) Version der zu filternden Ressource.
- **ResourceMatchers[].Namen**: (*Optional*) Namen im Kubernetes metadata.name-Feld der zu filternden Ressource.
- **ResourceMatchers[].Namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der zu filternden Ressource.
- **ResourceMatchers[].labelSelectors**: (*Optional*) Label selector string im Feld Kubernetes metadata.name der Ressource, wie im definiert "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

  Beispiel:

```
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. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-backup-ipr-cr.yaml` haben, wenden Sie den CR an:

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

Verwenden Sie die CLI

Schritte

1. Stellen Sie das Backup auf den ursprünglichen Namespace wieder her, und ersetzen Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung. Das `backup` Argument verwendet einen Namespace und einen Backup-Namen im Format `<namespace>/<name>`. Beispiel:

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

Wiederherstellung von einem Backup in einem anderen Cluster

Sie können ein Backup in einem anderen Cluster wiederherstellen, wenn ein Problem mit dem ursprünglichen Cluster auftritt.

Wenn Sie Backups mit Kopia als Datenmigrationsprogramm wiederherstellen, können Sie optional Anmerkungen in der CR oder über die CLI angeben, um das Verhalten des von Kopia verwendeten temporären Speichers zu steuern. Siehe die "Kopia-Dokumentation" Weitere Informationen zu den konfigurierbaren Optionen finden Sie hier. Verwenden Sie die `tridentctl-protect create --help` Befehl für weitere Informationen zum Festlegen von Annotationen mit der Trident Protect CLI.

Bevor Sie beginnen

Stellen Sie sicher, dass die folgenden Voraussetzungen erfüllt sind:

- Auf dem Zielcluster ist Trident Protect installiert.
- Der Zielcluster hat Zugriff auf den Bucket-Pfad desselben AppVault wie das Quellcluster, in dem das Backup gespeichert ist.
- Stellen Sie sicher, dass der Ablauf des AWS-Sitzungstokens für alle Wiederherstellungsvorgänge mit langer Laufzeit ausreicht. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.
  - Weitere Informationen zum Überprüfen des Ablaufes des aktuellen Sitzungstokens finden Sie im "AWS API-Dokumentation".
  - Weitere Informationen zu Zugangsdaten für AWS Ressourcen finden Sie in der "AWS-Dokumentation".

Schritte

1. Überprüfen Sie die Verfügbarkeit des AppVault CR auf dem Zielcluster mithilfe des Trident Protect CLI-Plugins:

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

   Stellen Sie sicher, dass der für die Anwendungswiederherstellung vorgesehene Namespace auf dem Zielcluster vorhanden ist.

2. Zeigen Sie die Backup-Inhalte des verfügbaren AppVault vom Zielcluster an:

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

Mit diesem Befehl werden die verfügbaren Backups im AppVault angezeigt, einschließlich der ursprünglichen Cluster, der entsprechenden Anwendungsnamen, Zeitstempel und Archivpfade.

Beispielausgabe:

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   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. Stellen Sie die Anwendung mithilfe des AppVault-Namens und Archivpfads auf dem Zielcluster wieder her:

CR verwenden

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-backup-restore-cr.yaml`.
2. Konfigurieren Sie in der erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*required*) der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **Spec.appVaultRef**: (*required*) der Name des AppVault, in dem der Backup-Inhalt gespeichert ist.
   - **Spec.appArchivePath**: Der Pfad innerhalb von AppVault, in dem die Backup-Inhalte gespeichert werden. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

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

     Wenn BackupRestore CR nicht verfügbar ist, können Sie den in Schritt 2 genannten Befehl verwenden, um den Inhalt des Backups anzuzeigen.

   - **spec.namespaceMapping**: die Zuordnung des Quell-Namespace des Wiederherstellungsvorgangs zum Ziel-Namespace. Ersetzen `my-source-namespace` Sie und `my-destination-namespace` mit Informationen aus Ihrer Umgebung.

     Beispiel:

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: my-cr-name
  namespace: my-destination-namespace
  annotations: # Optional annotations for Kopia data mover
    protect.trident.netapp.io/kopia-content-cache-size-limit-mb:
"1000"
spec:
  appVaultRef: appvault-name
  appArchivePath: my-backup-path
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-backup-restore-cr.yaml` haben, wenden Sie den CR an:

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

Verwenden Sie die CLI

1. Verwenden Sie den folgenden Befehl, um die Anwendung wiederherzustellen und Werte in Klammern durch Informationen aus Ihrer Umgebung zu ersetzen. Das Namespace-Mapping-Argument verwendet durch Doppelpunkte getrennte Namespaces, um Quellnamepaces im Format source1:dest1,source2:dest2 den korrekten Zielnamepaces zuzuordnen. Beispiel:

```
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>
```

Wiederherstellung von einem Snapshot in einem anderen Namespace

Sie können Daten aus einem Snapshot mithilfe einer benutzerdefinierten Ressourcendatei (CR-Datei) entweder in einem anderen Namensraum oder im ursprünglichen Quellnamensraum wiederherstellen. Wenn Sie einen Snapshot mithilfe eines SnapshotRestore CR in einem anderen Namespace wiederherstellen, stellt Trident Protect die Anwendung in einem neuen Namespace wieder her und erstellt einen Anwendungs-CR für die wiederhergestellte Anwendung. Um die wiederhergestellte Anwendung zu schützen, erstellen Sie bei Bedarf Backups oder Snapshots oder legen Sie einen Schutzzeitplan fest.

Bevor Sie beginnen

Vergewissern Sie sich, dass der Ablauf des AWS-Sitzungstokens für alle langwierigen s3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Überprüfen des Ablaufes des aktuellen Sitzungstokens finden Sie im "AWS API-Dokumentation".
- Weitere Informationen zu Zugangsdaten für AWS Ressourcen finden Sie in der "AWS IAM-Dokumentation".

CR verwenden

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-snapshot-restore-cr.yaml`.
2. Konfigurieren Sie in der erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*required*) der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **Spec.appVaultRef**: (*required*) der Name des AppVault, in dem der Snapshot-Inhalt gespeichert ist.
   - **Spec.appArchivePath**: Der Pfad innerhalb von AppVault, wo der Snapshot-Inhalt gespeichert wird. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

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

   - **spec.namespaceMapping**: die Zuordnung des Quell-Namespace des Wiederherstellungsvorgangs zum Ziel-Namespace. Ersetzen `my-source-namespace` Sie und `my-destination-namespace` mit Informationen aus Ihrer Umgebung.
   - **Spec.storageClassMapping**: Das Mapping der Quellspeicherklasse des Wiederherstellungsvorgangs an die Zielspeicherklasse. Ersetzen `destinationStorageClass` Sie und `sourceStorageClass` mit Informationen aus Ihrer Umgebung.

     > Der `storageClassMapping` Attribut funktioniert nur, wenn sowohl das ursprüngliche als auch das neue `StorageClass` Verwenden Sie dasselbe Speicher-Backend. Wenn Sie versuchen, auf einem `StorageClass` das ein anderes Speicher-Backend verwendet, schlägt der Wiederherstellungsvorgang fehl.

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

3. (*Optional*) Wenn Sie nur bestimmte Ressourcen der wiederherzustellenden Anwendung auswählen müssen, fügen Sie eine Filterung hinzu, die Ressourcen mit bestimmten Bezeichnungen enthält oder ausschließt:

   > Trident Protect wählt einige Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn Sie beispielsweise eine persistente Volume-Claim-Ressource auswählen und diese einen zugehörigen Pod hat, stellt Trident Protect auch den zugehörigen Pod wieder her.

   - **ResourceFilter.resourceSelectionCriteria**: (Erforderlich für die Filterung) Verwenden `Include` oder `Exclude` um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die einzuschließenden oder auszuschließenden Ressourcen zu definieren:
     - **RefindeFilter.refindeMatchers**: Eine Reihe von refindeMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, stimmen sie als OP-Operation überein, und die Felder innerhalb jedes Elements (Gruppe, Typ, Version) stimmen mit einer UND-Operation überein.
       - **ResourceMatchers[].Group**: (*Optional*) Gruppe der zu filternden Ressource.
       - **ResourceMatchers[].Kind**: (*Optional*) Art der zu filternden Ressource.
       - **ResourceMatchers[].Version**: (*Optional*) Version der zu filternden Ressource.
       - **ResourceMatchers[].Namen**: (*Optional*) Namen im Kubernetes metadata.name-Feld der zu filternden Ressource.
       - **ResourceMatchers[].Namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der zu filternden Ressource.
       - **ResourceMatchers[].labelSelectors**: (*Optional*) Label selector string im Feld Kubernetes metadata.name der Ressource, wie im definiert "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

         Beispiel:

```
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. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-snapshot-restore-cr.yaml` haben, wenden Sie den CR an:

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

Verwenden Sie die CLI

Schritte

1. Stellen Sie den Snapshot in einem anderen Namespace wieder her und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung.

   - Das `snapshot` Argument verwendet einen Namespace und Snapshot-Namen im Format `<namespace>/<name>`.
   - Das `namespace-mapping` Argument verwendet durch Doppelpunkte getrennte Namespaces, um Quellnamepaces im Format den richtigen Zielnamepaces zuzuordnen `source1:dest1,source2:dest2`.

     Beispiel:

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

Wiederherstellung von einem Snapshot im ursprünglichen Namespace

Sie können einen Snapshot jederzeit im ursprünglichen Namespace wiederherstellen.

Bevor Sie beginnen

Vergewissern Sie sich, dass der Ablauf des AWS-Sitzungstokens für alle langwierigen s3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Überprüfen des Ablaufes des aktuellen Sitzungstokens finden Sie im "AWS API-Dokumentation".
- Weitere Informationen zu Zugangsdaten für AWS Ressourcen finden Sie in der "AWS IAM-Dokumentation".

CR verwenden

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-snapshot-ipr-cr.yaml`.
2. Konfigurieren Sie in der erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*required*) der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **Spec.appVaultRef**: (*required*) der Name des AppVault, in dem der Snapshot-Inhalt gespeichert ist.
   - **Spec.appArchivePath**: Der Pfad innerhalb von AppVault, wo der Snapshot-Inhalt gespeichert wird. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

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

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

3. (*Optional*) Wenn Sie nur bestimmte Ressourcen der wiederherzustellenden Anwendung auswählen müssen, fügen Sie eine Filterung hinzu, die Ressourcen mit bestimmten Bezeichnungen enthält oder ausschließt:

   > Trident Protect wählt einige Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn Sie beispielsweise eine persistente Volume-Claim-Ressource auswählen und diese einen zugehörigen Pod hat, stellt Trident Protect auch den zugehörigen Pod wieder her.

   - **ResourceFilter.resourceSelectionCriteria**: (Erforderlich für die Filterung) Verwenden `Include` oder `Exclude` um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die einzuschließenden oder auszuschließenden Ressourcen zu definieren:
     - **RefindeFilter.refindeMatchers**: Eine Reihe von refindeMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, stimmen sie als OP-Operation überein, und die Felder innerhalb jedes Elements (Gruppe, Typ, Version) stimmen mit einer UND-Operation überein.
       - **ResourceMatchers[].Group**: (*Optional*) Gruppe der zu filternden Ressource.
       - **ResourceMatchers[].Kind**: (*Optional*) Art der zu filternden Ressource.

- **ResourceMatchers[].Version**: (*Optional*) Version der zu filternden Ressource.
- **ResourceMatchers[].Namen**: (*Optional*) Namen im Kubernetes metadata.name-Feld der zu filternden Ressource.
- **ResourceMatchers[].Namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der zu filternden Ressource.
- **ResourceMatchers[].labelSelectors**: (*Optional*) Label selector string im Feld Kubernetes metadata.name der Ressource, wie im definiert "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

  Beispiel:

```
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. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-snapshot-ipr-cr.yaml` haben, wenden Sie den CR an:

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

Verwenden Sie die CLI

Schritte

1. Stellen Sie den Snapshot auf den ursprünglichen Namespace wieder her, und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung. Beispiel:

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

Überprüfen Sie den Status eines Wiederherstellungsvorgangs

Sie können die Befehlszeile verwenden, um den Status eines Wiederherstellungsvorgangs zu überprüfen, der gerade ausgeführt wird, abgeschlossen wurde oder fehlgeschlagen ist.

Schritte

1. Verwenden Sie den folgenden Befehl, um den Status des Wiederherstellungsvorgangs abzurufen und Werte in Bracken durch Informationen aus Ihrer Umgebung zu ersetzen:

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

Anwendungen mit NetApp SnapMirror und Trident Protect replizieren

Mit Trident Protect können Sie die asynchronen Replikationsfunktionen der NetApp SnapMirror Technologie nutzen, um Daten und Anwendungsänderungen von einem Speichersystem auf ein anderes zu replizieren, entweder innerhalb desselben Clusters oder zwischen verschiedenen Clustern.

Namespace-Annotationen und -Labels bei Restore- und Failover-Vorgängen

Während von Restore- und Failover-Vorgängen werden Labels und Annotationen im Ziel-Namespace so angefertigt, dass sie den Labels und Annotationen im Quell-Namespace entsprechen. Beschriftungen oder Anmerkungen aus dem Quell-Namespace, die nicht im Ziel-Namespace vorhanden sind, werden hinzugefügt. Alle bereits vorhandenen Beschriftungen oder Anmerkungen werden überschrieben, um dem Wert aus dem Quell-Namespace zu entsprechen. Beschriftungen oder Anmerkungen, die nur im Ziel-Namespace vorhanden sind, bleiben unverändert.

> Wenn Sie Red hat OpenShift verwenden, ist es wichtig, die wichtige Rolle von Namespace-Annotationen in OpenShift-Umgebungen zu beachten. Namespace-Anmerkungen stellen sicher, dass wiederhergestellte Pods den durch OpenShift Security Context Constraints (SCCs) definierten Berechtigungen und Sicherheitskonfigurationen entsprechen und ohne Berechtigungsprobleme auf Volumes zugreifen können. Weitere Informationen finden Sie im "Dokumentation zu den Einschränkungen für den Sicherheitskontext von OpenShift".

Sie können verhindern, dass bestimmte Annotationen im Ziel-Namespace überschrieben werden, indem Sie die Kubernetes-Umgebungsvariable vor der Wiederherstellung oder dem Failover einstellen `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`. Beispiel:

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

Wenn Sie die Quellanwendung mit Helm installiert haben `--create-namespace` Flagge, besondere

Behandlung wird der `name` Legende mit Beschriftung. Während des Wiederherstellungs- oder Failover-Prozesses kopiert Trident Protect diese Bezeichnung in den Ziel-Namespace, aktualisiert aber den Wert auf den Wert des Ziel-Namespace, wenn der Wert aus der Quelle mit dem Quell-Namespace übereinstimmt. Stimmt dieser Wert nicht mit dem Quell-Namespace überein, wird er unverändert in den Ziel-Namespace kopiert.

Beispiel

Im folgenden Beispiel wird ein Quell- und Ziel-Namespace präsentiert, der jeweils unterschiedliche Annotationen und Labels enthält. Sie können den Status des Ziel-Namespace vor und nach dem Vorgang anzeigen und sehen, wie die Annotationen und Labels im Ziel-Namespace kombiniert oder überschrieben werden.

Vor der Wiederherstellung oder dem Failover

Die folgende Tabelle zeigt den Status der Beispiel-Namespaces für Quelle und Ziel vor dem Restore- oder Failover-Vorgang:

Namespace	Anmerkungen	Etiketten
Namespace ns-1 (Quelle)	• Annotation.one/key: „Aktualisiertwert“ • Anmerkung.zwei/Taste: "Wahr"	• Umgebung = Produktion • Compliance=hipaa • Name=ns-1
Namespace ns-2 (Ziel)	• Anmerkung.One/key: „True“ • Anmerkung.drei/Taste: "False"	• Rolle=Datenbank

Nach dem Wiederherstellungsvorgang

In der folgenden Tabelle wird der Status des Beispiel-Ziel-Namespace nach der Wiederherstellung oder dem Failover-Vorgang dargestellt. Einige Schlüssel wurden hinzugefügt, einige wurden überschrieben und das Label wurde aktualisiert, `name` um dem Ziel-Namespace zu entsprechen:

Namespace	Anmerkungen	Etiketten
Namespace ns-2 (Ziel)	• Annotation.one/key: „Aktualisiertwert“ • Anmerkung.zwei/Taste: "Wahr" • Anmerkung.drei/Taste: "False"	• Name=ns-2 • Compliance=hipaa • Umgebung = Produktion • Rolle=Datenbank

Sie können Trident Protect so konfigurieren, dass Dateisysteme während Datensicherungsvorgängen eingefroren und wieder freigegeben werden. "Erfahren Sie mehr über die Konfiguration des Einfrierens von Dateisystemen mit Trident Protect.".

Richten Sie eine Replikationsbeziehung ein

Die Einrichtung einer Replikationsbeziehung umfasst Folgendes:

- Auswahl der Häufigkeit, mit der Trident Protect einen App-Snapshot erstellen soll (der sowohl die

Kubernetes-Ressourcen der App als auch die Volume-Snapshots für jedes der App-Volumes umfasst).

- Auswahl des Replizierungszeitplans (einschließlich Kubernetes-Ressourcen und persistenter Volume-Daten)
- Einstellen der Uhrzeit für die Erstellung des Snapshots

Schritte

1. Erstellen Sie im Quellcluster einen AppVault für die Quellanwendung. Ändern Sie abhängig von Ihrem Storage-Anbieter ein Beispiel in "Benutzerdefinierte Ressourcen von AppVault"entsprechend Ihrer Umgebung:

Erstellen Sie einen AppVault mit einem CR

a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (z. B. `trident-protect-appvault-primary-source.yaml`).

b. Konfigurieren Sie die folgenden Attribute:

- **metadata.name**: (*required*) der Name der benutzerdefinierten AppVault-Ressource. Notieren Sie sich den ausgewählten Namen, da sich andere CR-Dateien, die für eine Replikationsbeziehung benötigt werden, auf diesen Wert beziehen.
- **spec.providerConfig**: (*required*) speichert die Konfiguration, die für den Zugriff auf den AppVault unter Verwendung des angegebenen Providers erforderlich ist. Wählen Sie einen BucketName und alle anderen notwendigen Details für Ihren Anbieter. Notieren Sie sich die ausgewählten Werte, da sich andere CR-Dateien, die für eine Replikationsbeziehung benötigt werden, auf diese Werte beziehen. Beispiele für AppVault CRS mit anderen Anbietern finden Sie unter"Benutzerdefinierte Ressourcen von AppVault".
- **spec.providerCredentials**: (*required*) speichert Verweise auf alle Anmeldeinformationen, die für den Zugriff auf AppVault unter Verwendung des angegebenen Anbieters erforderlich sind.
  - **spec.providerCredentials.valueFromSecret**: (*required*) gibt an, dass der Wert der Zugangsdaten von einem Secret stammen soll.
    - **Key**: (*required*) der gültige Schlüssel des zu wählenden Geheimnisses.
    - **Name**: (*required*) Name des Geheimnisses, das den Wert für dieses Feld enthält. Muss sich im gleichen Namespace befinden.
  - **spec.providerCredentials.secretAccessKey**: (*required*) der Zugriffsschlüssel, der für den Zugriff auf den Provider verwendet wird. Der **Name** sollte mit **spec.providerCredentials.valueFromSecret.name** übereinstimmen.
- **spec.providerType**: (*required*) legt fest, was das Backup bereitstellt, z. B. NetApp ONTAP S3, generisches S3, Google Cloud oder Microsoft Azure. Mögliche Werte:
  - AWS
  - Azure
  - GCP
  - Generisch-s3
  - ONTAP-s3
  - StorageGRID-s3

c. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-appvault-primary-source.yaml` haben, wenden Sie den CR an:

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

Erstellen Sie einen AppVault mithilfe der CLI

a. Erstellen Sie AppVault und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung:

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

2. Erstellen Sie auf dem Quellcluster die CR-Quellanwendung:

Erstellen Sie die Quellanwendung mit einem CR

a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (z. B. `trident-protect-app-source.yaml`).

b. Konfigurieren Sie die folgenden Attribute:

- **metadata.name**: (*required*) der Name der benutzerdefinierten Ressource der Anwendung. Notieren Sie sich den ausgewählten Namen, da sich andere CR-Dateien, die für eine Replikationsbeziehung benötigt werden, auf diesen Wert beziehen.
- **spec.includedNamespaces**: (*required*) ein Array von Namespaces und zugehörigen Labels. Verwenden Sie Namespace-Namen und Grenzen Sie optional den Umfang der Namespaces mit Beschriftungen ein, um Ressourcen anzugeben, die in den hier aufgeführten Namespaces vorhanden sind. Der Application Namespace muss Teil dieses Arrays sein.

  Beispiel 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. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-app-source.yaml` haben, wenden Sie den CR an:

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

Erstellen Sie die Quellanwendung mithilfe der CLI

a. Erstellen Sie die Quellanwendung. Beispiel:

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

3. Optional können Sie auf dem Quellcluster einen Snapshot zum Herunterfahren der Quellanwendung erstellen. Dieser Snapshot wird als Basis für die Anwendung auf dem Zielcluster verwendet. Wenn Sie diesen Schritt überspringen, müssen Sie warten, bis der nächste geplante Snapshot ausgeführt wird, damit Sie über einen aktuellen Snapshot verfügen.

Erstellen Sie einen Snapshot zum Herunterfahren mit einem CR

a. Erstellen Sie einen Replikationszeitplan für die Quellanwendung:

i. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (z. B. `trident-protect-schedule.yaml`).

ii. Konfigurieren Sie die folgenden Attribute:

- **metadata.name**: (*required*) der Name der benutzerdefinierten Ressource für den Zeitplan.
- **Spec.AppVaultRef**: (*required*) dieser Wert muss mit dem Feld metadata.name des AppVault für die Quellanwendung übereinstimmen.
- **Spec.ApplicationRef**: (*required*) dieser Wert muss mit dem Feld metadata.name der Quellanwendung CR übereinstimmen.
- **Spec.backupRetention**: (*required*) Dieses Feld ist erforderlich und der Wert muss auf 0 gesetzt werden.
- **Spec.enabled**: Muss auf true gesetzt werden.
- **spec.granularity**: muss auf eingestellt sein `Custom`.
- **Spec.recurrenceRule**: Definieren Sie ein Startdatum in UTC-Zeit und ein Wiederholungsintervall.
- **Spec.snapshotRetention**: Muss auf 2 gesetzt werden.

Beispiel YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule-0e1f88ab-f013-4bce-8ae9-6afed9df59a1
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

i. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-schedule.yaml` haben, wenden Sie den CR an:

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

Erstellen Sie einen Snapshot zum Herunterfahren mit der CLI

a. Erstellen Sie den Snapshot, und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung. Beispiel:

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

4. Erstellen Sie auf dem Zielcluster eine Quellanwendung AppVault CR, die mit der AppVault CR identisch ist, die Sie auf das Quellcluster angewendet haben, und benennen Sie sie (z. B. `trident-protect-appvault-primary-destination.yaml`).
5. CR anwenden:

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

6. Erstellen Sie ein Ziel-AppVault CR für die Zielanwendung auf dem Zielcluster. Ändern Sie abhängig von Ihrem Storage-Anbieter ein Beispiel in "Benutzerdefinierte Ressourcen von AppVault"entsprechend Ihrer Umgebung:
   a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (z. B. `trident-protect-appvault-secondary-destination.yaml`).
   b. Konfigurieren Sie die folgenden Attribute:
      - **metadata.name**: (*required*) der Name der benutzerdefinierten AppVault-Ressource. Notieren Sie sich den ausgewählten Namen, da sich andere CR-Dateien, die für eine Replikationsbeziehung benötigt werden, auf diesen Wert beziehen.
      - **spec.providerConfig**: (*required*) speichert die Konfiguration, die für den Zugriff auf den AppVault unter Verwendung des angegebenen Providers erforderlich ist. Wählen Sie eine `bucketName` und alle anderen notwendigen Details für Ihren Provider. Notieren Sie sich die ausgewählten Werte, da sich andere CR-Dateien, die für eine Replikationsbeziehung benötigt werden, auf diese Werte beziehen. Beispiele für AppVault CRS mit anderen Anbietern finden Sie unter"Benutzerdefinierte Ressourcen von AppVault".
      - **spec.providerCredentials**: (*required*) speichert Verweise auf alle Anmeldeinformationen, die für den Zugriff auf AppVault unter Verwendung des angegebenen Anbieters erforderlich sind.
        - **spec.providerCredentials.valueFromSecret**: (*required*) gibt an, dass der Wert der Zugangsdaten von einem Secret stammen soll.
          - **Key**: (*required*) der gültige Schlüssel des zu wählenden Geheimnisses.
          - **Name**: (*required*) Name des Geheimnisses, das den Wert für dieses Feld enthält. Muss sich im gleichen Namespace befinden.

        - **spec.providerCredentials.secretAccessKey**: (*required*) der Zugriffsschlüssel, der für den Zugriff auf den Provider verwendet wird. Der **Name** sollte mit **spec.providerCredentials.valueFromSecret.name** übereinstimmen.
    - **spec.providerType**: (*required*) legt fest, was das Backup bereitstellt, z. B. NetApp ONTAP S3, generisches S3, Google Cloud oder Microsoft Azure. Mögliche Werte:
        - AWS
        - Azure
        - GCP
        - Generisch-s3
        - ONTAP-s3
        - StorageGRID-s3

c. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-appvault-secondary-destination.yaml` haben, wenden Sie den CR an:

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

7. Erstellen Sie auf dem Zielcluster eine AppMirrorRelationship CR-Datei:

Erstellen Sie eine AppMirrorRelationship mithilfe eines CR

a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (z. B. `trident-protect-relationship.yaml`).

b. Konfigurieren Sie die folgenden Attribute:

- **metadata.name:** (erforderlich) der Name der benutzerdefinierten AppMirrorRelationship-Ressource.
- **spec.destinationAppVaultRef**: (*required*) dieser Wert muss mit dem Namen des AppVault für die Zielanwendung auf dem Zielcluster übereinstimmen.
- **spec.namespaceMapping**: (*required*) der Ziel- und Quellnamepaces muss mit dem im jeweiligen Anwendungs-CR definierten AnwendungsNamespace übereinstimmen.
- **Spec.sourceAppVaultRef**: (*required*) dieser Wert muss mit dem Namen des AppVault für die Quellanwendung übereinstimmen.
- **Spec.sourceApplicationName**: (*required*) dieser Wert muss mit dem Namen der Quellanwendung übereinstimmen, die Sie in der Quellanwendung CR definiert haben.
- **Spec.storageClassName**: (*required*) Wählen Sie den Namen einer gültigen Storage-Klasse auf dem Cluster. Die Storage-Klasse muss mit einer ONTAP Storage-VM verknüpft werden, die mit der Quellumgebung peert wird.
- **Spec.recurrenceRule**: Definieren Sie ein Startdatum in UTC-Zeit und ein Wiederholungsintervall.

  Beispiel 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. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-relationship.yaml` haben, wenden Sie den CR an:

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

Erstellen Sie eine AppMirrorRelationship mithilfe der CLI

a. Erstellen und wenden Sie das AppMirrorRelationship-Objekt an, und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung. Beispiel:

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

8. (*Optional*) Überprüfen Sie auf dem Zielcluster den Status und den Status der Replikationsbeziehung:

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

Failover zum Ziel-Cluster

Mit Trident Protect können Sie replizierte Anwendungen auf einen Zielcluster verschieben, ohne dass es zu einem Failover kommt. Dieses Verfahren beendet die Replikationsbeziehung und schaltet die Anwendung auf dem Zielcluster online. Trident Protect stoppt die Anwendung auf dem Quellcluster nicht, wenn sie betriebsbereit war.

Schritte

1. Bearbeiten Sie im Zielcluster die AppMirrorRelationship CR-Datei (z. B. `trident-protect-relationship.yaml`) und ändern Sie den Wert von **spec.desiredState** in `Promoted`.
2. Speichern Sie die CR-Datei.
3. CR anwenden:

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

4. (*Optional*) Erstellen Sie alle Schutzzeitpläne, die Sie für die Failover-Anwendung benötigen.
5. (*Optional*) Status und Status der Replikationsbeziehung prüfen:

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

Neusynchronisierung einer fehlgeschlagenen Replikationsbeziehung

Durch den Neusynchronisierung wird die Replikationsbeziehung wiederhergestellt. Nach einer Neusynchronisierung wird die ursprüngliche Quellanwendung zur laufenden Anwendung, und alle Änderungen, die an der laufenden Anwendung auf dem Zielcluster vorgenommen werden, werden verworfen.

Der Prozess stoppt die App auf dem Zielcluster, bevor die Replikation wiederhergestellt wird.

> Alle Daten, die während des Failovers auf die Zielapplikation geschrieben werden, gehen verloren.

Schritte

1. Optional: Erstellen Sie auf dem Quellcluster einen Snapshot der Quellanwendung. Dadurch wird sichergestellt, dass die neuesten Änderungen vom Quellcluster erfasst werden.
2. Bearbeiten Sie im Zielcluster die AppMirrorRelationship CR-Datei (z. B. `trident-protect-relationship.yaml`) und ändern Sie den Wert spec.desiredState in `Established`.
3. Speichern Sie die CR-Datei.
4. CR anwenden:

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

5. Wenn Sie Schutzzeitpläne auf dem Zielcluster erstellt haben, um die Failover-Anwendung zu schützen, entfernen Sie sie. Alle Zeitpläne, die weiterhin zu Fehlern bei Volume-Snapshots führen.

Reverse Resynchronisierung einer Failover-Replizierungsbeziehung

Wenn Sie eine Failover-Replikationsbeziehung umkehren, wird die Zielanwendung zur Quellanwendung, und die Quelle wird zum Ziel. Änderungen, die während des Failovers an der Zielapplikation vorgenommen werden, werden beibehalten.

Schritte

1. Löschen Sie auf dem ursprünglichen Zielcluster die AppMirrorRelationship-CR. Dadurch wird das Ziel zur Quelle. Wenn auf dem neuen Ziel-Cluster noch Schutzpläne vorhanden sind, entfernen Sie sie.
2. Richten Sie eine Replikationsbeziehung ein, indem Sie die CR-Dateien anwenden, die Sie ursprünglich zum Einrichten der Beziehung zu den anderen Clustern verwendet haben.
3. Stellen Sie sicher, dass das neue Ziel (ursprünglicher Quellcluster) mit beiden AppVault CRS konfiguriert ist.
4. Richten Sie eine Replikationsbeziehung auf dem anderen Cluster ein, und konfigurieren Sie Werte für die umgekehrte Richtung.

Richtung der Anwendungsreplikation umkehren

Wenn Sie die Replikationsrichtung umkehren, verschiebt Trident Protect die Anwendung zum Ziel-Speicher-

Backend, während die Replikation zurück zum ursprünglichen Quell-Speicher-Backend fortgesetzt wird. Trident Protect stoppt die Quellanwendung und repliziert die Daten zum Ziel, bevor ein Failover zur Zielanwendung erfolgt.

In dieser Situation tauschen Sie Quelle und Ziel aus.

Schritte

1. Erstellen Sie auf dem Quellcluster einen Snapshot zum Herunterfahren:

Erstellen Sie einen Snapshot zum Herunterfahren mit einem CR

a. Deaktivieren Sie die Schutzrichtlinienpläne für die Quellanwendung.

b. Erstellen Sie eine CR-Datei für den ShutdownSnapshot:

   i. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (z. B. `trident-protect-shutdownsnapshot.yaml`).

   ii. Konfigurieren Sie die folgenden Attribute:

   - **metadata.name**: (*required*) der Name der benutzerdefinierten Ressource.
   - **Spec.AppVaultRef**: (*required*) dieser Wert muss mit dem Feld metadata.name des AppVault für die Quellanwendung übereinstimmen.
   - **Spec.ApplicationRef**: (*required*) dieser Wert muss mit dem Feld metadata.name der CR-Datei der Quellanwendung übereinstimmen.

     Beispiel 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. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-shutdownsnapshot.yaml` haben, wenden Sie den CR an:

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

Erstellen Sie einen Snapshot zum Herunterfahren über die CLI

a. Erstellen Sie den Snapshot zum Herunterfahren, und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung. Beispiel:

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

2. Rufen Sie auf dem Quellcluster nach Abschluss des Snapshot zum Herunterfahren den Status des Snapshot zum Herunterfahren ab:

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

3. Suchen Sie im Quellcluster den Wert von **shutdownSnapshot.Status.appArchivePath** mit dem folgenden Befehl und notieren Sie den letzten Teil des Dateipfads (auch Basisname genannt; dies ist alles nach dem letzten Schrägstrich):

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

4. Führen Sie mit der folgenden Änderung ein Failover vom neuen Zielcluster zum neuen Quellcluster durch:

> Fügen Sie in Schritt 2 des Failover-Verfahrens das Feld in die AppMirrorRelationship CR-Datei ein `spec.promotedSnapshot`, und setzen Sie den Wert auf den Basisnamen, den Sie oben in Schritt 3 aufgezeichnet haben.

5. Führen Sie die Schritte für die umgekehrte Resynchronisierung in Reverse Resynchronisierung einer Failover-Replizierungsbeziehungaus.
6. Aktivieren Sie Schutzzeitpläne auf dem neuen Quellcluster.

Ergebnis

Die folgenden Aktionen werden aufgrund der umgekehrten Replikation durchgeführt:

- Von den Kubernetes-Ressourcen der ursprünglichen Quell-Applikation wird ein Snapshot erstellt.
- Die PODs der ursprünglichen Quell-App werden mit sanfter Weise gestoppt, indem die Kubernetes-Ressourcen der App gelöscht werden (wodurch PVCs und PVS aktiviert bleiben).
- Nach dem Herunterfahren der Pods werden Snapshots der Volumes der App erstellt und repliziert.
- Die SnapMirror Beziehungen sind beschädigt, wodurch die Zieldatenträger für Lese-/Schreibvorgänge bereit sind.
- Die Kubernetes-Ressourcen der App werden aus dem Snapshot vor dem Herunterfahren wiederhergestellt. Dabei werden die Volume-Daten verwendet, die nach dem Herunterfahren der ursprünglichen Quell-App repliziert wurden.
- Die Replizierung wird in umgekehrter Richtung wieder hergestellt.

Führen Sie ein Failback von Anwendungen auf das ursprüngliche Quellcluster durch

Mit Trident Protect können Sie nach einem Failover-Vorgang ein „Failback“ erreichen, indem Sie die folgende Abfolge von Operationen verwenden. In diesem Workflow zur Wiederherstellung der ursprünglichen Replikationsrichtung repliziert (resynchronisiert) Trident Protect alle Anwendungsänderungen zurück in die ursprüngliche Quellanwendung, bevor die Replikationsrichtung umgekehrt wird.

Dieser Prozess beginnt mit einer Beziehung, bei der ein Failover zu einem Ziel durchgeführt wurde, und umfasst die folgenden Schritte:

- Starten Sie mit einem Failover-Status fehlgeschlagen.
- Umgekehrte Neusynchronisierung der Replikationsbeziehung.

> Führen Sie keine normale Neusynchronisierung durch, da dadurch während des Failover Daten, die auf das Ziel-Cluster geschrieben werden, verworfen werden.

- Kehren Sie die Replikationsrichtung um.

Schritte

1. Führen Sie die Reverse Resynchronisierung einer Failover-Replizierungsbeziehung Schritte aus.
2. Führen Sie die Richtung der Anwendungsreplikation umkehren Schritte aus.

Löschen einer Replikationsbeziehung

Sie können eine Replikationsbeziehung jederzeit löschen. Wenn Sie die Anwendungsreplikationsbeziehung löschen, führt dies zu zwei separaten Anwendungen ohne Beziehung zwischen ihnen.

Schritte

1. Löschen Sie auf dem aktuellen Desination-Cluster die CR-Datei AppMirrorRelationship:

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

Anwendungen mit Trident Protect migrieren

Sie können Ihre Applikationen zwischen Clustern oder Storage-Klassen migrieren, indem Sie Ihre Backup- oder Snapshot-Daten in einem anderen Cluster oder einer anderen Storage-Klasse wiederherstellen.

> Wenn Sie eine Anwendung migrieren, werden alle für die Anwendung konfigurierten Ausführungshaken mit der App migriert. Wenn ein Hook für die Ausführung nach der Wiederherstellung vorhanden ist, wird er automatisch als Teil des Wiederherstellungsvorgangs ausgeführt.

Backup- und Restore-Vorgänge

Um Backup- und Wiederherstellungsvorgänge für die folgenden Szenarien durchzuführen, können Sie bestimmte Backup- und Wiederherstellungsaufgaben automatisieren.

Klonen auf dasselbe Cluster

Um eine Anwendung auf demselben Cluster zu klonen, erstellen Sie einen Snapshot oder ein Backup, und stellen Sie die Daten im selben Cluster wieder her.

Schritte

1. Führen Sie einen der folgenden Schritte aus:
   a. "Erstellen Sie einen Snapshot".
   b. "Erstellen Sie ein Backup".

2. Führen Sie auf demselben Cluster je nach Erstellung eines Snapshots oder eines Backups einen der folgenden Schritte aus:
   a. "Stellen Sie Ihre Daten aus dem Snapshot wieder her".
   b. "Stellen Sie Ihre Daten aus dem Backup wieder her".

Klonen auf anderes Cluster

Um eine Anwendung auf einen anderen Cluster zu klonen (clusterübergreifendes Klonen), erstellen Sie eine Sicherung auf dem Quellcluster und stellen Sie diese Sicherung anschließend auf dem anderen Cluster wieder her. Stellen Sie sicher, dass Trident Protect auf dem Zielcluster installiert ist.

> Sie können eine Anwendung zwischen verschiedenen Clustern mit replizieren"SnapMirror Replizierung".

Schritte

1. "Erstellen Sie ein Backup".
2. Stellen Sie sicher, dass der AppVault CR für den Objektspeicher-Bucket, der das Backup enthält, auf dem Zielcluster konfiguriert wurde.
3. Auf dem Zielcluster, "Stellen Sie Ihre Daten aus dem Backup wieder her".

Migrieren von Applikationen von einer Storage-Klasse zu einer anderen Storage-Klasse

Sie können Anwendungen von einer Storage-Klasse zu einer anderen Storage-Klasse migrieren, indem Sie einen Snapshot auf der anderen Ziel-Storage-Klasse wiederherstellen.

Beispiel: (Ohne die Geheimnisse aus dem Wiederherstellungs-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
```

Stellen Sie den Snapshot mithilfe eines CR-Systems wieder her

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-snapshot-restore-cr.yaml`.
2. Konfigurieren Sie in der erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*required*) der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **Spec.appArchivePath**: Der Pfad innerhalb von AppVault, wo der Snapshot-Inhalt gespeichert wird. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

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

   - **Spec.appVaultRef**: (*required*) der Name des AppVault, in dem der Snapshot-Inhalt gespeichert ist.
   - **spec.namespaceMapping**: die Zuordnung des Quell-Namespace des Wiederherstellungsvorgangs zum Ziel-Namespace. Ersetzen `my-source-namespace` Sie und `my-destination-namespace` mit Informationen aus Ihrer Umgebung.

     ```
     ---
     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. Wenn Sie optional nur bestimmte Ressourcen der wiederherzustellenden Anwendung auswählen müssen, fügen Sie eine Filterung hinzu, die Ressourcen mit bestimmten Bezeichnungen enthält oder ausschließt:
   - **ResourceFilter.resourceSelectionCriteria**: (Erforderlich für die Filterung) Verwenden Sie `include or exclude`, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die einzuschließenden oder auszuschließenden Ressourcen zu definieren:
     - **RefindeFilter.refindeMatchers**: Eine Reihe von refindeMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, stimmen sie als OP-Operation überein, und die Felder innerhalb jedes Elements (Gruppe, Typ, Version) stimmen mit einer UND-Operation überein.
       - **ResourceMatchers[].Group**: (*Optional*) Gruppe der zu filternden Ressource.
       - **ResourceMatchers[].Kind**: (*Optional*) Art der zu filternden Ressource.

- **ResourceMatchers[].Version**: (*Optional*) Version der zu filternden Ressource.
- **ResourceMatchers[].Namen**: (*Optional*) Namen im Kubernetes metadata.name-Feld der zu filternden Ressource.
- **ResourceMatchers[].Namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der zu filternden Ressource.
- **ResourceMatchers[].labelSelectors**: (*Optional*) Label selector string im Feld Kubernetes metadata.name der Ressource, wie im definiert "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

  Beispiel:

```
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. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-snapshot-restore-cr.yaml` haben, wenden Sie den CR an:

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

Stellen Sie den Snapshot mithilfe der CLI wieder her

Schritte

1. Stellen Sie den Snapshot in einem anderen Namespace wieder her und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung.

   - Das `snapshot` Argument verwendet einen Namespace und Snapshot-Namen im Format `<namespace>/<name>`.
   - Das `namespace-mapping` Argument verwendet durch Doppelpunkte getrennte Namespaces, um Quellnamepaces im Format den richtigen Zielnamepaces zuzuordnen `source1:dest1,source2:dest2`.

     Beispiel:

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

Trident Protect-Ausführungshooks verwalten

Ein Execution Hook ist eine benutzerdefinierte Aktion, die Sie so konfigurieren können, dass sie zusammen mit einem Datenschutzvorgang einer verwalteten App ausgeführt wird. Wenn Sie beispielsweise über eine Datenbank-App verfügen, können Sie mit einem Execution-Hook alle Datenbanktransaktionen vor einem Snapshot anhalten und die Transaktionen nach Abschluss des Snapshots wieder aufnehmen. Dies gewährleistet applikationskonsistente Snapshots.

Arten von Ausführungshaken

Trident Protect unterstützt die folgenden Arten von Ausführungs-Hooks, je nachdem, wann sie ausgeführt werden können:

- Vor dem Snapshot
- Nach dem Snapshot
- Vor dem Backup
- Nach dem Backup
- Nach dem Wiederherstellen
- Nach Failover

Ausführungsreihenfolge

Wenn ein Datenschutzvorgang ausgeführt wird, finden Hakenereignisse in der folgenden Reihenfolge statt:

1. Alle entsprechenden benutzerdefinierten Testhaken für die Ausführung vor dem Betrieb werden auf den entsprechenden Containern ausgeführt. Sie können beliebig viele benutzerdefinierte Hooks für die Vorbedienung erstellen und ausführen, aber die Reihenfolge der Ausführung dieser Haken vor der Operation ist weder garantiert noch konfigurierbar.
2. Gegebenenfalls kommt es zum Einfrieren des Dateisystems. "Erfahren Sie mehr über die Konfiguration des Einfrierens von Dateisystemen mit Trident Protect.".
3. Der Vorgang der Datensicherung wird durchgeführt.
4. Eingefrorene Dateisysteme werden gegebenenfalls nicht eingefroren.
5. Alle entsprechenden benutzerdefinierten Testhaken für die Ausführung nach der Operation werden auf den entsprechenden Containern ausgeführt. Sie können beliebig viele benutzerdefinierte Haken für die Nachbearbeitung erstellen und ausführen, aber die Reihenfolge der Ausführung dieser Haken nach der Operation ist weder garantiert noch konfigurierbar.

Wenn Sie mehrere Testausführungshaken desselben Typs erstellen (z. B. Pre-Snapshot), ist die Reihenfolge der Ausführung dieser Haken nicht garantiert. Die Reihenfolge der Ausführung von Haken unterschiedlicher

Art ist jedoch garantiert. Im Folgenden wird beispielsweise die Reihenfolge der Ausführung einer Konfiguration beschrieben, die alle verschiedenen Hooks umfasst:

1. Hooks vor dem Snapshot wurden ausgeführt
2. Hooks nach dem Snapshot wurden ausgeführt
3. Hooks vor dem Backup wurden ausgeführt
4. Hooks nach dem Backup ausgeführt

Das Beispiel der vorherigen Reihenfolge gilt nur, wenn Sie ein Backup ausführen, das keinen vorhandenen Snapshot verwendet.

Sie sollten Ihre Hook-Skripte immer testen, bevor Sie sie in einer Produktionsumgebung aktivieren. Mit dem Befehl 'kubectl exec' können Sie die Skripte bequem testen. Nachdem Sie die Testausführungshaken in einer Produktionsumgebung aktiviert haben, testen Sie die erstellten Snapshots und Backups, um sicherzustellen, dass sie konsistent sind. Dazu klonen Sie die Applikation in einem temporären Namespace, stellen den Snapshot oder das Backup wieder her und testen anschließend die App.

Wenn ein Hook aus einer Phase vor der Snapshot-Ausführung Kubernetes-Ressourcen hinzufügt, ändert oder entfernt, werden diese Änderungen im Snapshot oder Backup und in jedem nachfolgenden Wiederherstellungsvorgang enthalten.

Wichtige Hinweise zu benutzerdefinierten Testausführungshaken

Bei der Planung von Testausführungshooks für Ihre Apps sollten Sie Folgendes berücksichtigen:

- Ein Testsuite muss ein Skript verwenden, um Aktionen durchzuführen. Viele Testsuitehaoks können auf dasselbe Skript verweisen.
- Trident Protect verlangt, dass die Skripte, die von den Ausführungs-Hooks verwendet werden, im Format ausführbarer Shell-Skripte geschrieben sind.
- Die Skriptgröße ist auf 96 KB begrenzt.
- Trident Protect verwendet die Einstellungen für Ausführungs-Hooks und alle übereinstimmenden Kriterien, um zu bestimmen, welche Hooks für einen Snapshot-, Backup- oder Wiederherstellungsvorgang anwendbar sind.

Da Testsuitehingel die Funktionalität der Anwendung, für die sie ausgeführt werden, oft reduzieren oder vollständig deaktivieren, sollten Sie immer versuchen, die Zeit zu minimieren, die Ihre benutzerdefinierten Testausführungshaken für die Ausführung benötigt. Wenn Sie eine Backup- oder Snapshot-Operation mit zugeordneten Testsuiten starten, diese aber dann abbrechen, können die Haken trotzdem ausgeführt werden, wenn der Backup- oder Snapshot-Vorgang bereits gestartet wurde. Das bedeutet, dass die in einem Testsuite nach dem Backup verwendete Logik nicht davon ausgehen kann, dass das Backup abgeschlossen wurde.

Filter für Testausführungshaken

Wenn Sie einen Ausführungshaken für eine Anwendung hinzufügen oder bearbeiten, können Sie dem Ausführungshaken Filter hinzufügen, um zu verwalten, welche Container der Hook entsprechen soll. Filter sind für Applikationen nützlich, die in allen Containern dasselbe Container-Image nutzen. Jedes Image kann jedoch für einen anderen Zweck (wie Elasticsearch) verwendet werden. Mit Filtern können Sie Szenarien erstellen, in

denen Ausführungshaken auf einigen, aber nicht unbedingt allen identischen Containern ausgeführt werden. Wenn Sie mehrere Filter für einen einzelnen Testausführungshaken erstellen, werden diese mit einem logischen UND einem Operator kombiniert. Pro Testsuite können Sie bis zu 10 aktive Filter haben.

Jeder Filter, den Sie einem Ausführungs-Hook hinzufügen, verwendet einen regulären Ausdruck, um Container in Ihrem Cluster abzugleichen. Wenn ein Hook mit einem Container übereinstimmt, führt der Hook das zugehörige Skript auf diesem Container aus. Reguläre Ausdrücke für Filter verwenden die Syntax „Regulärer Ausdruck 2“ (RE2), die das Erstellen eines Filters, der Container aus der Liste der Übereinstimmungen ausschließt, nicht unterstützt. Informationen zur Syntax, die Trident Protect für reguläre Ausdrücke in Ausführungs-Hook-Filtern unterstützt, finden Sie unter "Syntaxunterstützung für regulären Ausdruck 2 (RE2)" Die

> Wenn Sie einem Ausführungs-Hook einen Namespace-Filter hinzufügen, der nach einer Wiederherstellung oder einem Klonvorgang ausgeführt wird, und die Wiederherstellungs- oder Klonquelle und das Ziel in verschiedenen Namespaces liegen, wird der Namespace-Filter nur auf den Ziel-Namespace angewendet.

Beispiele für Testausführungshaken

Besuchen Sie die "NetApp Verda GitHub Projekt" , um echte Ausführungshaken für gängige Apps wie Apache Cassandra und Elasticsearch herunterzuladen. Sie können auch Beispiele sehen und Ideen für die Strukturierung Ihrer eigenen benutzerdefinierten Execution Hooks erhalten.

Erstellen Sie einen Ausführungshaken

Sie können mit Trident Protect einen benutzerdefinierten Ausführungs-Hook für eine App erstellen. Sie benötigen die Berechtigungen Eigentümer, Administrator oder Mitglied, um Ausführungs-Hooks zu erstellen.

CR verwenden

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-hook.yaml`.
2. Konfigurieren Sie die folgenden Attribute so, dass sie Ihrer Trident Protect-Umgebung und Clusterkonfiguration entsprechen:
   - **metadata.name**: (*required*) der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **Spec.applicationRef**: (*required*) der Kubernetes-Name der Anwendung, für die der Ausführungshaken ausgeführt werden soll.
   - **Spec.Stage**: (*required*) Eine Zeichenfolge, die angibt, welche Phase während der Aktion der Ausführungshaken ausgeführt werden soll. Mögliche Werte:
     - Vor
     - Post
   - **Spec.Action**: (*required*) Eine Zeichenfolge, die angibt, welche Aktion der Ausführungshaken ausführen wird, vorausgesetzt, dass alle angegebenen Ausführungshaken-Filter übereinstimmen. Mögliche Werte:
     - Snapshot
     - Backup
     - Wiederherstellen
     - Failover
   - **Spec.enabled**: (*Optional*) gibt an, ob dieser Ausführungshaken aktiviert oder deaktiviert ist. Wenn nicht angegeben, ist der Standardwert TRUE.
   - **Spec.hookSource**: (*required*) Ein String, der das base64-kodierte Hook-Skript enthält.
   - **Spec.timeout**: (*Optional*) Eine Zahl, die definiert, wie lange der Ausführungshaken in Minuten ausgeführt werden darf. Der Mindestwert beträgt 1 Minute, und der Standardwert ist 25 Minuten, wenn nicht angegeben.
   - **Spec.Arguments**: (*Optional*) Eine YAML-Liste von Argumenten, die Sie für den Ausführungshaken angeben können.
   - **Spec.matchingCriteria**: (*Optional*) eine optionale Liste von Kriterien-Schlüsselwertpaaren, jedes Paar, das einen Ausführungshook-Filter bildet. Sie können bis zu 10 Filter pro Ausführungshaken hinzufügen.
   - **Spec.matchingCriteria.type**: (*Optional*) Eine Zeichenfolge, die den Filtertyp für den Ausführungshaken identifiziert. Mögliche Werte:
     - ContainerImage
     - Containername
     - PodName
     - PodLabel
     - NamespaceName
   - **Spec.matchingCriteria.value**: (*Optional*) Ein String oder regulärer Ausdruck, der den Wert des Ausführungshook-Filters identifiziert.

Beispiel 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. Nachdem Sie die CR-Datei mit den richtigen Werten ausgefüllt haben, wenden Sie den CR an:

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

Verwenden Sie die CLI

Schritte

1. Erstellen Sie den Ausführungshaken, und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung. Beispiel:

```
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>
```

Führen Sie manuell einen Ausführungshaken aus

Sie können einen Ausführungshaken manuell zu Testzwecken ausführen oder den Hook nach einem Fehler manuell erneut ausführen. Sie müssen über die Berechtigungen Eigentümer, Administrator oder Mitglied verfügen, um Ausführungshaken manuell auszuführen.

Das manuelle Ausführen eines Ausführungshakens besteht aus zwei grundlegenden Schritten:

1. Erstellen Sie ein Ressourcenbackup, das Ressourcen sammelt und eine Sicherung von ihnen erstellt und bestimmt, wo der Hook ausgeführt wird
2. Führen Sie den Ausführungshaken gegen die Sicherung aus

Schritt 1: Erstellen einer Ressourcensicherung

CR verwenden

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-resource-backup.yaml`.
2. Konfigurieren Sie die folgenden Attribute so, dass sie Ihrer Trident Protect-Umgebung und Clusterkonfiguration entsprechen:
   - **metadata.name**: (*required*) der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **Spec.applicationRef**: (*required*) der Kubernetes-Name der Applikation, für die das Ressourcen-Backup erstellt werden soll.
   - **Spec.appVaultRef**: (*required*) der Name des AppVault, in dem der Backup-Inhalt gespeichert ist.
   - **Spec.appArchivePath**: Der Pfad innerhalb von AppVault, in dem die Backup-Inhalte gespeichert werden. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

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

     Beispiel 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. Nachdem Sie die CR-Datei mit den richtigen Werten ausgefüllt haben, wenden Sie den CR an:

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

Verwenden Sie die CLI

Schritte

1. Erstellen Sie das Backup, und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung. Beispiel:

```
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. Den Status des Backups anzeigen. Sie können diesen Beispielbefehl wiederholt verwenden, bis der Vorgang abgeschlossen ist:

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

3. Überprüfen Sie, ob die Sicherung erfolgreich war:

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

Schritt 2: Führen Sie den Ausführungshaken aus

CR verwenden

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-hook-run.yaml`.
2. Konfigurieren Sie die folgenden Attribute so, dass sie Ihrer Trident Protect-Umgebung und Clusterkonfiguration entsprechen:

   - **metadata.name**: (*required*) der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **Spec.applicationRef**: (*required*) Stellen Sie sicher, dass dieser Wert mit dem Anwendungsnamen aus dem ResourceBackup CR übereinstimmt, den Sie in Schritt 1 erstellt haben.
   - **Spec.appVaultRef**: (*required*) Stellen Sie sicher, dass dieser Wert mit der appVaultRef aus der ResourceBackup CR übereinstimmt, die Sie in Schritt 1 erstellt haben.
   - **Spec.appArchivePath**: Stellen Sie sicher, dass dieser Wert mit dem appArchivePath aus dem ResourceBackup CR übereinstimmt, den Sie in Schritt 1 erstellt haben.

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

   - **Spec.Action**: (*required*) Eine Zeichenfolge, die angibt, welche Aktion der Ausführungshaken ausführen wird, vorausgesetzt, dass alle angegebenen Ausführungshaken-Filter übereinstimmen. Mögliche Werte:
     - Snapshot
     - Backup
     - Wiederherstellen
     - Failover
   - **Spec.Stage**: (*required*) Eine Zeichenfolge, die angibt, welche Phase während der Aktion der Ausführungshaken ausgeführt werden soll. Bei diesem Hakenlauf werden keine Haken in einer anderen Phase ausgeführt. Mögliche Werte:
     - Vor
     - Post

     Beispiel 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. Nachdem Sie die CR-Datei mit den richtigen Werten ausgefüllt haben, wenden Sie den CR an:

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

Verwenden Sie die CLI

Schritte

1. Erstellen Sie die Anforderung zur manuellen Ausführung von Hook Run:

```
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. Überprüfen Sie den Status des Ausführungs-Hook-Durchlaufs. Sie können diesen Befehl wiederholt ausführen, bis der Vorgang abgeschlossen ist:

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

3. Beschreiben Sie das exechoksrun-Objekt, um die endgültigen Details und den Status anzuzeigen:

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