Anwendungen verwalten und schützen

Trident

NetApp
March 05, 2026

Inhalt

Anwendungen verwalten und schützen

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 eine AppVault-CR manuell oder über die Befehlszeile erstellen, wenn Sie Datenschutzvorgänge für eine Anwendung ausführen. Die AppVault-CR ist spezifisch für Ihre Umgebung. Die Beispiele auf dieser Seite können Ihnen beim Erstellen von AppVault-CRs als Leitfaden dienen.

> Stellen Sie sicher, dass sich der AppVault CR auf dem Cluster befindet, auf dem Trident Protect installiert ist. Wenn der AppVault CR nicht vorhanden ist oder Sie nicht darauf zugreifen können, wird in der Befehlszeile ein Fehler angezeigt.

AppVault-Authentifizierung und Passwörter konfigurieren

Bevor Sie einen AppVault CR erstellen, stellen Sie sicher, dass sich der AppVault und der von Ihnen gewählte Datenverschieber beim Anbieter und allen zugehörigen Ressourcen authentifizieren können.

Passwörter für das Datenübertragungs-Repository

Wenn Sie AppVault-Objekte mithilfe von CRs oder dem Trident Protect CLI-Plugin erstellen, können Sie ein Kubernetes-Secret mit benutzerdefinierten Passwörtern für die Restic- und Kopia-Verschlüsselung angeben. Wenn Sie kein Geheimnis angeben, verwendet Trident Protect ein Standardpasswort.

- Beim manuellen Erstellen von AppVault CRs 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 ein Passwort für das Datenübertragungs-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 Datenmigrationsdienst Sie verwenden, müssen Sie nur das entsprechende Passwort für diesen Datenmigrationsdienst angeben. Wenn Sie beispielsweise Restic verwenden und nicht planen, Kopia in Zukunft zu nutzen, können Sie beim Erstellen des Geheimnisses nur das Restic-Passwort angeben.
> - Bewahren Sie das Passwort an einem sicheren Ort auf. Sie benötigen es, um Daten auf demselben oder einem anderen Cluster wiederherzustellen. Wenn der Cluster oder der `trident-protect` Wenn der Namespace gelöscht wird, können Sie Ihre Backups oder Snapshots ohne das Passwort nicht wiederherstellen.

Verwenden Sie einen CR

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

Verwenden der 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 Abschnitt „…“. "Amazon S3-Dokumentation" .

EKS Pod Identity für Amazon S3 (AWS)-Authentifizierung

Trident Protect unterstützt EKS Pod Identity für Kopia-Datenmigrationsvorgänge. Diese Funktion ermöglicht den sicheren Zugriff auf S3-Buckets, ohne AWS-Anmeldeinformationen in Kubernetes-Geheimnissen zu speichern.

Anforderungen für EKS Pod Identity mit Trident Protect

Bevor Sie EKS Pod Identity mit Trident Protect verwenden, stellen Sie Folgendes sicher:

- In Ihrem EKS-Cluster ist Pod Identity aktiviert.
- Sie haben eine IAM-Rolle mit den erforderlichen S3-Bucket-Berechtigungen erstellt. Weitere Informationen finden Sie unter "S3-kompatible Speicher-IAM-Berechtigungen".
- Die IAM-Rolle ist den folgenden Trident Protect-Dienstkonten zugeordnet:
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

Detaillierte Anweisungen zum Aktivieren von Pod Identity und zum Zuordnen von IAM-Rollen zu Servicekonten finden Sie in der Dokumentation. "AWS EKS Pod Identity-Dokumentation" .

AppVault-Konfiguration Wenn Sie EKS Pod Identity verwenden, konfigurieren Sie Ihre AppVault CR mit der `useIAM: true` Flagge statt expliziter Anmeldeinformationen:

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

AppVault-Schlüsselgenerierungsbeispiele für Cloud-Anbieter

Wenn Sie eine AppVault CR definieren, müssen Sie Anmeldeinformationen für den Zugriff auf die vom Anbieter gehosteten Ressourcen angeben, es sei denn, Sie verwenden die IAM-Authentifizierung. Die Art und Weise, wie Sie die Schlüssel für die Zugangsdaten generieren, unterscheidet sich je nach Anbieter. Nachfolgend finden Sie Beispiele für die Schlüsselgenerierung über die Befehlszeile für mehrere Anbieter. Sie

können die folgenden Beispiele verwenden, um Schlüssel für die Anmeldeinformationen jedes Cloud-Anbieters zu 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
```

Generisches 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

Nachfolgend finden Sie Beispieldefinitionen für AppVault für jeden Anbieter.

AppVault CR-Beispiele

Anhand der folgenden CR-Beispiele können Sie AppVault-Objekte für jeden Cloud-Anbieter erstellen.

- Optional können Sie ein Kubernetes-Secret angeben, das benutzerdefinierte Passwörter für die Verschlüsselung der Restic- und Kopia-Repositorys enthält. SiehePasswörter für das Datenübertragungs-Repository für weitere Informationen.
- Bei Amazon S3 (AWS) AppVault-Objekten können Sie optional ein SessionToken angeben, was nützlich ist, wenn Sie Single Sign-On (SSO) zur Authentifizierung verwenden. Dieses Token wird erstellt, wenn Sie Schlüssel für den Anbieter generieren.AppVault-Schlüsselgenerierungsbeispiele für Cloud-Anbieter .
- Für S3 AppVault-Objekte können Sie optional eine Egress-Proxy-URL für ausgehenden S3-Datenverkehr angeben. `spec.providerConfig.S3.proxyURL` Schlüssel.

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
```

Bei EKS-Umgebungen, die Pod Identity mit Kopia Data Mover verwenden, können Sie die `providerCredentials` Abschnitt und hinzufügen `useIAM: true` unter dem `s3` stattdessen Konfiguration.

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
```

Generisches 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

```yaml
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.

- Optional können Sie ein Kubernetes-Secret angeben, das benutzerdefinierte Passwörter für die Verschlüsselung der Restic- und Kopia-Repositorys enthält. SiehePasswörter für das Datenübertragungs-Repository für weitere Informationen.
- Für S3 AppVault-Objekte können Sie optional eine Egress-Proxy-URL für ausgehenden S3-Datenverkehr angeben. `--proxy-url <ip_address:port>` Argument.

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
```

Generisches 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
```

AppVault-Informationen anzeigen

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

Schritte

1. Den 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. Optional kann der AppVaultPath für jede Ressource mit dem Flag angezeigt werden. `--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` .

AppVault entfernen

Sie können ein AppVault-Objekt jederzeit entfernen.

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

Bevor Sie beginnen

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

Entfernen eines AppVaults mithilfe der Kubernetes-CLI

1. Entfernen Sie das AppVault-Objekt und ersetzen Sie es durch `appvault-name` mit dem 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 es durch `appvault-name` mit dem 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 eine 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"AppVault-Benutzerressourcen."

Definieren einer 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 mithilfe eines CR hinzu

Schritte

1. Erstellen Sie die CR-Datei der Zielanwendung:
   a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) und benennen Sie sie (zum Beispiel). `maria-app.yaml`).
   b. Konfigurieren Sie die folgenden Attribute:
      - **metadata.name**: (*Erforderlich*) Der Name der benutzerdefinierten Anwendungsressource. Merken Sie sich den gewählten Namen, da andere für Schutzvorgänge benötigte CR-Dateien auf diesen Wert verweisen.
      - **spec.includedNamespaces**: (*Erforderlich*) Verwenden Sie den Namespace- und Label-Selektor, um die Namespaces und Ressourcen anzugeben, die die Anwendung verwendet. Der Anwendungs-Namespace muss Teil dieser Liste sein. Der Label-Selektor ist optional und kann verwendet werden, um Ressourcen innerhalb jedes angegebenen Namespace zu filtern.
      - **spec.includedClusterScopedResources**: (*Optional*) Verwenden Sie dieses Attribut, um Cluster-bezogene Ressourcen anzugeben, die in die Anwendungsdefinition aufgenommen werden sollen. Mithilfe dieses Attributs können Sie diese Ressourcen anhand ihrer Gruppe, Version, Art und Bezeichnungen auswählen.
        - **groupVersionKind**: (*Erforderlich*) Gibt die API-Gruppe, Version und Art der clusterweiten Ressource an.
        - **labelSelector**: (*Optional*) Filtert die clusterweiten Ressourcen anhand ihrer Labels.
      - **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" .

Falls Sie diese Annotation nachträglich anwenden müssen, nachdem eine Anwendung bereits erstellt wurde, können Sie 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. (*Optional*) Filter hinzufügen, die Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließen:

   - **resourceFilter.resourceSelectionCriteria**: (Für die Filterung erforderlich) Verwenden Sie `Include` oder `Exclude` Eine in resourceMatchers definierte Ressource ein- oder ausschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die ein- oder auszuschließenden Ressourcen zu definieren:

     - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese per OR-Verknüpfung abgeglichen, und die Felder innerhalb jedes Elements (Gruppe, Art, Version) werden per AND-Verknüpfung abgeglichen.
       - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressourcen.
       - **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
       - **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.

- **resourceMatchers[].names**: (*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-Selektorzeichenfolge im Kubernetes-Metadatenfeld „name“ der Ressource, wie in der Dokumentation definiert. "Kubernetes-Dokumentation" . Zum Beispiel: `"trident.netapp.io/os=linux"` .

> Wenn beide `resourceFilter` Und `labelSelector` werden verwendet `resourceFilter` zuerst läuft und dann `labelSelector` wird auf die resultierenden Ressourcen angewendet.

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"]
```

2. Nachdem Sie die Anwendungs-CR erstellt haben, die zu Ihrer Umgebung passt, wenden Sie die CR an. Beispiel:

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

Schritte

1. Erstellen und wenden Sie die Anwendungsdefinition anhand eines der folgenden Beispiele an und ersetzen Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung. Sie können Namespaces und Ressourcen in die Anwendungsdefinition einbinden, indem Sie kommagetrennte Listen mit den in den Beispielen gezeigten Argumenten verwenden.

   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 Annotation bei der Erstellung einer Anwendung über die CLI anzugeben, können Sie Folgendes verwenden: `--annotation` Flagge.

- Erstellen Sie die Anwendung und verwenden Sie die globale Einstellung für das Einfrieren des Dateisystems:

```
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 lokalen Anwendungseinstellungen für das Dateisystem-Einfrierverhalten:

```
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">
```

Sie können verwenden `--resource-filter-include` Und `--resource-filter-exclude` Kennzeichnungen zum Ein- oder Ausschließen von Ressourcen basierend auf `resourceSelectionCriteria` wie beispielsweise Gruppe, Art, Version, Bezeichnungen, Namen und Namensräume, wie im folgenden Beispiel gezeigt:

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

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.

Clusterbezogene Ressourcen werden in eine Sicherung, einen Snapshot oder einen Klon aufgenommen, wenn sie in der Anwendungsdefinition explizit referenziert werden oder wenn sie Verweise auf einen der Anwendungs-Namespaces enthalten.

Erstellen Sie einen Snapshot mithilfe eines CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) und benennen Sie sie. `trident-protect-snapshot-cr.yaml` .
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und aussagekräftigen Namen für Ihre Umgebung.
   - **spec.applicationRef**: Der Kubernetes-Name der Anwendung, für die ein Snapshot erstellt werden soll.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVaults, in dem die Snapshot-Inhalte (Metadaten) gespeichert werden sollen.
   - **spec.reclaimPolicy**: (*Optional*) Definiert, was mit dem AppArchive eines Snapshots geschieht, wenn der Snapshot CR gelöscht wird. Das bedeutet, dass selbst wenn auf eingestellt `Retain` Der Snapshot wird gelöscht. 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 `trident-protect-snapshot-cr.yaml` Datei mit den korrekten Werten, CR anwenden:

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

Erstellen Sie einen Snapshot mithilfe der Befehlszeile.

Schritte

1. Erstellen Sie den Snapshot und ersetzen Sie die 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 eine bedarfsgesteuerte Datensicherung.

Sie können eine App jederzeit sichern.

> Clusterbezogene Ressourcen werden in eine Sicherung, einen Snapshot oder einen Klon aufgenommen, wenn sie in der Anwendungsdefinition explizit referenziert werden oder wenn sie Verweise auf einen der Anwendungs-Namespaces enthalten.

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger laufenden S3-Backup-Vorgänge ausreichend ist. Wenn das Token während des Sicherungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Siehe die "AWS API-Dokumentation" Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie hier.
- Siehe die "AWS IAM-Dokumentation" Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie hier.

Erstellen Sie eine Sicherung mithilfe eines CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) und benennen Sie sie. `trident-protect-backup-cr.yaml` .
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:

   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und aussagekräftigen Namen für Ihre Umgebung.
   - **spec.applicationRef**: (*Erforderlich*) Der Kubernetes-Name der zu sichernden Anwendung.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVaults, in dem die Sicherungsinhalte gespeichert werden sollen.
   - **spec.dataMover**: (*Optional*) Eine Zeichenkette, die angibt, welches Sicherungstool für den Sicherungsvorgang verwendet werden soll. Mögliche Werte (Groß-/Kleinschreibung beachten):
     - `Restic`
     - `Kopia`(Standard)
   - **spec.reclaimPolicy**: (*Optional*) Definiert, was mit einem Backup geschieht, wenn es aus seinem Anspruch freigegeben wird. Mögliche Werte:
     - `Delete`
     - `Retain`(Standard)
   - **spec.snapshotRef**: (*Optional*): Name des Snapshots, der als Quelle für die Sicherung verwendet werden soll. Falls keine Daten angegeben werden, wird ein temporärer Snapshot erstellt und gesichert.
   - **metadata.annotations.protect.trident.netapp.io/full-backup** : (*Optional*) Mit dieser Annotation kann festgelegt werden, ob ein Backup nicht inkrementell sein soll. Standardmäßig sind alle Sicherungen inkrementell. Wenn diese Annotation jedoch auf „“ gesetzt ist, … `true` , die Datensicherung wird nicht mehr inkrementell. Sofern nicht anders angegeben, erfolgt die Datensicherung gemäß der Standardeinstellung für inkrementelle Datensicherung. Es empfiehlt sich, regelmäßig eine vollständige Datensicherung durchzuführen und zwischen den vollständigen Datensicherungen inkrementelle Datensicherungen vorzunehmen, um das mit der Wiederherstellung verbundene Risiko zu minimieren.

     Beispiel-YAML:

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

3. Nachdem Sie die `trident-protect-backup-cr.yaml` Datei mit den korrekten Werten, CR anwenden:

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

Erstellen Sie ein Backup mithilfe der Befehlszeile.

Schritte

1. Erstellen Sie die Sicherungskopie und ersetzen Sie die 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>
```

   Sie können optional die `--full-backup` Kennzeichen zur Angabe, ob eine Sicherung nicht inkrementell sein soll. Standardmäßig sind alle Sicherungen inkrementell. Wenn diese Option aktiviert ist, wird die Datensicherung nicht inkrementell durchgeführt. Es empfiehlt sich, regelmäßig eine vollständige Datensicherung durchzuführen und zwischen den vollständigen Datensicherungen inkrementelle Datensicherungen vorzunehmen, um das mit der Wiederherstellung verbundene Risiko zu minimieren.

Erstellen Sie einen Datenschutzplan

Eine Schutzrichtlinie schützt eine App, indem sie nach einem festgelegten Zeitplan Snapshots, Backups oder beides erstellt. Sie können stündlich, täglich, wöchentlich und monatlich Snapshots und Backups erstellen und die Anzahl der aufzubewahrenden Kopien angeben. Sie können eine nicht inkrementelle vollständige Sicherung planen, indem Sie die Annotation „full-backup-rule“ verwenden. Standardmäßig sind alle Sicherungen inkrementell. Durch regelmäßiges Durchführen einer vollständigen Sicherung und inkrementeller Sicherungen zwischendurch können Sie das mit Wiederherstellungen verbundene Risiko verringern.

- Zeitpläne für Snapshots können nur durch die folgende Einstellung erstellt werden: `backupRetention` auf Null und `snapshotRetention` auf einen Wert größer als Null. Einstellung `snapshotRetention` Der Wert Null bedeutet, dass bei geplanten Backups weiterhin Snapshots erstellt werden, diese jedoch nur temporär sind und unmittelbar nach Abschluss des Backups gelöscht werden.
- Clusterbezogene Ressourcen werden in eine Sicherung, einen Snapshot oder einen Klon aufgenommen, wenn sie in der Anwendungsdefinition explizit referenziert werden oder wenn sie Verweise auf einen der Anwendungs-Namespaces enthalten.

Erstellen Sie einen Zeitplan mithilfe eines CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) und benennen Sie sie. `trident-protect-schedule-cr.yaml` .
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und aussagekräftigen Namen für Ihre Umgebung.
   - **spec.dataMover**: (*Optional*) Eine Zeichenkette, die angibt, welches Sicherungstool für den Sicherungsvorgang verwendet werden soll. Mögliche Werte (Groß-/Kleinschreibung beachten):
     - `Restic`
     - `Kopia`(Standard)
   - **spec.applicationRef**: Der Kubernetes-Name der zu sichernden Anwendung.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVaults, in dem die Sicherungsinhalte gespeichert werden sollen.
   - **spec.backupRetention**: Die Anzahl der aufzubewahrenden Backups. Null gibt an, dass keine Sicherungen erstellt werden sollen (nur Snapshots).
   - **spec.snapshotRetention**: Die Anzahl der aufzubewahrenden Snapshots. Null bedeutet, dass keine Snapshots erstellt werden sollen.
   - **spec.granularity**: Die Häufigkeit, mit der der Zeitplan ausgeführt werden soll. Mögliche Werte und zugehörige Pflichtfelder:
     - `Hourly`(erfordert, dass Sie angeben) `spec.minute`)
     - `Daily`(erfordert, dass Sie angeben) `spec.minute` Und `spec.hour`)
     - `Weekly`(erfordert, dass Sie angeben) `spec.minute, spec.hour` , Und `spec.dayOfWeek`)
     - `Monthly`(erfordert, dass Sie angeben) `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 „“ eingestellt ist. `Monthly` . Der Wert muss als Zeichenfolge angegeben werden.
   - **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 „“ eingestellt ist. `Weekly` . Der Wert muss als Zeichenfolge angegeben werden.
   - **spec.hour**: (*Optional*) Die Stunde des Tages (0 - 23), zu der der Zeitplan ausgeführt werden soll. Dieses Feld ist erforderlich, wenn die Granularität auf „“ eingestellt ist. `Daily` , `Weekly` , oder `Monthly` . Der Wert muss als Zeichenfolge angegeben werden.
   - **spec.minute**: (*Optional*) Die Minute der Stunde (0 – 59), zu der der Zeitplan ausgeführt werden soll. Dieses Feld ist erforderlich, wenn die Granularität auf „“ eingestellt ist. `Hourly` , `Daily` , `Weekly` , oder `Monthly` . Der Wert muss als Zeichenfolge angegeben werden.
   - **metadata.annotations.protect.trident.netapp.io/full-backup-rule**: (*Optional*) Diese Annotation dient zur Angabe der Regel für die Planung einer vollständigen Datensicherung. Sie können es

einstellen auf `always` für eine kontinuierliche vollständige Datensicherung oder zur individuellen Anpassung an Ihre Bedürfnisse. Wenn Sie beispielsweise die tägliche Granularität wählen, können Sie die Wochentage festlegen, an denen eine vollständige Datensicherung erfolgen soll.

Beispiel-YAML für Sicherungs- und Snapshot-Zeitplan:

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

Beispiel-YAML für einen Nur-Snapshot-Zeitplan:

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

3. Nachdem Sie die `trident-protect-schedule-cr.yaml` Datei mit den korrekten Werten, CR anwenden:

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

Erstellen Sie einen Zeitplan mithilfe der Befehlszeile.

Schritte

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

   Sie können verwenden `tridentctl-protect create schedule --help` Um detaillierte Hilfeinformationen zu diesem Befehl anzuzeigen.

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

   Sie können die `--full-backup-rule` Flagge an `always` für eine kontinuierliche vollständige Datensicherung oder zur individuellen Anpassung an Ihre Bedürfnisse. Wenn Sie beispielsweise die tägliche Granularität wählen, können Sie die Wochentage festlegen, an denen die vollständige Datensicherung erfolgen soll. Verwenden Sie beispielsweise `--full-backup-rule "Monday,Thursday"` Um eine vollständige Datensicherung montags und donnerstags einzuplanen.

   Für Zeitpläne, die nur Momentaufnahmen enthalten, stellen Sie Folgendes ein: `--backup -retention 0` und geben Sie einen Wert größer als 0 an für `--snapshot-retention` .

Löschen eines Snapshots

Löschen Sie die geplanten oder bedarfsgesteuerten Snapshots, die Sie nicht mehr benötigen.

Schritte

1. Entfernen Sie die dem Snapshot zugeordnete Snapshot-CR:

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

Sicherung löschen

Löschen Sie die geplanten oder bedarfsgesteuerten Backups, die Sie nicht mehr benötigen.

Stellen Sie sicher, dass die Rückforderungsrichtlinie auf Folgendes eingestellt ist: `Delete` Alle Sicherungsdaten aus dem Objektspeicher entfernen. Die Standardeinstellung der Richtlinie ist `Retain` um versehentlichen Datenverlust zu vermeiden. Wenn die Richtlinie nicht geändert wird `Delete` Die Sicherungsdaten bleiben im Objektspeicher erhalten und müssen manuell gelöscht werden.

Schritte

1. Entfernen Sie die zum Backup verknüpfte Sicherungs-CR:

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

Überprüfen Sie den Status eines Sicherungsvorgangs

Über die Befehlszeile können Sie den Status eines laufenden, abgeschlossenen oder fehlgeschlagenen Sicherungsvorgangs überprüfen.

Schritte

1. Verwenden Sie den folgenden Befehl, um den Status des Sicherungsvorgangs abzurufen. Ersetzen Sie dabei die Werte in Klammern durch Informationen aus Ihrer Umgebung:

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

Sichern und Wiederherstellen für Azure-NetApp-Dateien (ANF)-Vorgänge aktivieren

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.
- Du hast `azure-netapp-files` Als Standard-Speicherklasse für Ihr Speicher-Backend ausgewählt.

Für Konfigurationsschritte erweitern

1. Führen Sie in Trident die folgenden Schritte aus, 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-files 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 zugehörige PV aktiviert wurde:

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

   Antwort:

   ```
   snapshotDirectory: "true"
   ```

   +
   Wenn das Snapshot-Verzeichnis nicht aktiviert ist, wird die reguläre Sicherungsfunktionalität verwendet, die während des Sicherungsvorgangs 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 wiederherstellen

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 Ausführungs-Hooks mit der Anwendung wiederhergestellt. Wenn ein Hook für die Nachbearbeitung der Wiederherstellung vorhanden ist, wird dieser automatisch im Rahmen des Wiederherstellungsvorgangs ausgeführt.
- Die Wiederherstellung aus einer Sicherung in einen anderen Namespace oder in den ursprünglichen Namespace wird für Qtree-Volumes unterstützt. Die Wiederherstellung von einem Snapshot in einen anderen Namespace oder in den ursprünglichen Namespace wird für Qtree-Volumes jedoch nicht unterstützt.
- Sie können erweiterte Einstellungen verwenden, um Wiederherstellungsvorgänge anzupassen. Weitere Informationen finden Sie unter "Verwenden Sie die erweiterten Trident Protect-Wiederherstellungseinstellungen.".

Wiederherstellung aus einem Backup in einen anderen Namensraum

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.

Das Wiederherstellen einer Sicherung in einem anderen Namensraum mit bereits vorhandenen Ressourcen ändert keine Ressourcen, die denselben Namen wie die Ressourcen in der Sicherung haben. Um alle Ressourcen im Backup wiederherzustellen, müssen Sie entweder den Ziel-Namespace löschen und neu erstellen oder das Backup in einem neuen Namespace wiederherstellen.

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger andauernden S3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Siehe die "AWS API-Dokumentation" Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie hier.
- Siehe die "AWS IAM-Dokumentation" Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie hier.

Wenn Sie Sicherungen mit Kopia als Datenverschieber wiederherstellen, können Sie optional Anmerkungen im CR angeben oder die CLI verwenden, 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.

Verwenden Sie einen CR

Schritte

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

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

   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVaults, in dem die Sicherungsinhalte gespeichert sind.
   - **spec.namespaceMapping**: Die Zuordnung des Quell-Namespace der Wiederherstellungsoperation zum Ziel-Namespace. Ersetzen `my-source-namespace` Und `my-destination-namespace` mit Informationen aus Ihrer Umgebung.

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

3. (*Optional*) Falls Sie nur bestimmte Ressourcen der Anwendung für die Wiederherstellung auswählen möchten, fügen Sie Filter hinzu, die Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließen:

   > 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**: (Für die Filterung erforderlich) Verwenden Sie `Include` oder `Exclude` Eine in resourceMatchers definierte Ressource ein- oder ausschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die ein- oder auszuschließenden Ressourcen zu definieren:

- **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese per OR-Verknüpfung abgeglichen, und die Felder innerhalb jedes Elements (Gruppe, Art, Version) werden per AND-Verknüpfung abgeglichen.
  - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressourcen.
  - **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
  - **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
  - **resourceMatchers[].names**: (*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-Selektorzeichenfolge im Kubernetes-Metadatenfeld „name“ der Ressource, wie in der Dokumentation 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 `trident-protect-backup-restore-cr.yaml` Datei mit den korrekten Werten, CR anwenden:

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

Verwenden der CLI

Schritte

1. Stellen Sie die Sicherung in einem anderen Namespace wieder her und ersetzen Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung. Der `namespace-mapping` Das Argument verwendet durch Doppelpunkte getrennte Namensräume, um Quell-Namensräume den korrekten

Ziel-Namensräumen im folgenden Format 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 aus einer Sicherung in den ursprünglichen Namensraum

Sie können eine Sicherung jederzeit im ursprünglichen Namensraum wiederherstellen.

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger andauernden S3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Siehe die "AWS API-Dokumentation" Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie hier.
- Siehe die "AWS IAM-Dokumentation" Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie hier.

Wenn Sie Sicherungen mit Kopia als Datenverschieber wiederherstellen, können Sie optional Anmerkungen im CR angeben oder die CLI verwenden, 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.

Verwenden Sie einen CR

Schritte

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

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

   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVaults, in dem die Sicherungsinhalte gespeichert sind.

     Beispiel:

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

3. (*Optional*) Falls Sie nur bestimmte Ressourcen der Anwendung für die Wiederherstellung auswählen möchten, fügen Sie Filter hinzu, die Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließen:

   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**: (Für die Filterung erforderlich) Verwenden Sie `Include` oder `Exclude` Eine in resourceMatchers definierte Ressource ein- oder ausschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die ein- oder auszuschließenden Ressourcen zu definieren:
     - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese per OR-Verknüpfung abgeglichen, und die Felder innerhalb jedes Elements (Gruppe, Art, Version) werden per AND-Verknüpfung abgeglichen.

- **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressourcen.
- **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
- **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
- **resourceMatchers[].names**: (*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-Selektorzeichenfolge im Kubernetes-Metadatenfeld „name“ der Ressource, wie in der Dokumentation 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 `trident-protect-backup-ipr-cr.yaml` Datei mit den korrekten Werten, CR anwenden:

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

Verwenden der CLI

Schritte

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

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

Wiederherstellung aus einem Backup auf einem anderen Cluster

Sie können eine Sicherung auf einem anderen Cluster wiederherstellen, falls es ein Problem mit dem ursprünglichen Cluster gibt.

> Wenn Sie Sicherungen mit Kopia als Datenverschieber wiederherstellen, können Sie optional Anmerkungen im CR angeben oder die CLI verwenden, 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 der Quellcluster, in dem das Backup gespeichert ist.
- Stellen Sie sicher, dass Ihre lokale Umgebung beim Ausführen des AppVault CR eine Verbindung zum im AppVault CR definierten Objektspeicher-Bucket herstellen kann. `tridentctl-protect get appvaultcontent` Befehl. Falls Netzwerkbeschränkungen den Zugriff verhindern, führen Sie die Trident Protect CLI stattdessen innerhalb eines Pods auf dem Zielcluster aus.
- Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger andauernden Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.
  - Siehe die "AWS API-Dokumentation" Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie hier.
  - Siehe die "AWS-Dokumentation" Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie hier.

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 Sicherungsinhalte des verfügbaren AppVaults vom Zielcluster an:

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

Durch Ausführen dieses Befehls werden die verfügbaren Backups im AppVault angezeigt, einschließlich ihrer Ursprungscluster, zugehörigen 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. Die Anwendung mithilfe des AppVault-Namens und des Archivpfads im Zielcluster wiederherstellen:

Verwenden Sie einen CR

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) und benennen Sie sie. `trident-protect-backup-restore-cr.yaml` .
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und aussagekräftigen Namen für Ihre Umgebung.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVaults, in dem die Sicherungsinhalte gespeichert sind.
   - **spec.appArchivePath**: Der Pfad innerhalb von AppVault, in dem die Sicherungsinhalte gespeichert sind. Sie können folgenden Befehl verwenden, um diesen Pfad zu finden:

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

     Falls BackupRestore CR nicht verfügbar ist, können Sie den in Schritt 2 genannten Befehl verwenden, um den Sicherungsinhalt anzuzeigen.

   - **spec.namespaceMapping**: Die Zuordnung des Quell-Namespace der Wiederherstellungsoperation zum Ziel-Namespace. Ersetzen `my-source-namespace` 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
spec:
  appVaultRef: appvault-name
  appArchivePath: my-backup-path
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. Nachdem Sie die `trident-protect-backup-restore-cr.yaml` Datei mit den korrekten Werten, CR anwenden:

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

Verwenden der CLI

1. Verwenden Sie den folgenden Befehl, um die Anwendung wiederherzustellen, und ersetzen Sie dabei die Werte in Klammern durch Informationen aus Ihrer Umgebung. Das Argument namespace-

mapping verwendet durch Doppelpunkte getrennte Namensräume, um Quell-Namensräume den richtigen Ziel-Namensräumen im Format source1:dest1,source2:dest2 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 aus einem Snapshot in einen 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.

SnapshotRestore unterstützt die `spec.storageClassMapping` Attribut, jedoch nur dann, wenn die Quell- und Zielspeicherklassen das gleiche Speicher-Backend verwenden. Wenn Sie versuchen, einen Wiederherstellungszustand wiederherzustellen `StorageClass` Wenn ein anderes Speichersystem verwendet wird, schlägt die Wiederherstellung fehl.

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger andauernden S3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Siehe die "AWS API-Dokumentation" Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie hier.
- Siehe die "AWS IAM-Dokumentation" Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie hier.

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) und benennen Sie sie. `trident-protect-snapshot-restore-cr.yaml` .
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und aussagekräftigen Namen für Ihre Umgebung.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVaults, in dem die Snapshot-Inhalte gespeichert sind.
   - **spec.appArchivePath**: Der Pfad innerhalb von AppVault, in dem die Snapshot-Inhalte gespeichert sind. Sie können 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 der Wiederherstellungsoperation zum Ziel-Namespace. Ersetzen `my-source-namespace` Und `my-destination-namespace` mit Informationen aus Ihrer Umgebung.

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

3. (*Optional*) Falls Sie nur bestimmte Ressourcen der Anwendung für die Wiederherstellung auswählen möchten, fügen Sie Filter hinzu, die Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließen:

   > 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**: (Für die Filterung erforderlich) Verwenden Sie `Include` oder `Exclude` Eine in resourceMatchers definierte Ressource ein- oder ausschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die ein- oder auszuschließenden Ressourcen zu definieren:

- **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese per OR-Verknüpfung abgeglichen, und die Felder innerhalb jedes Elements (Gruppe, Art, Version) werden per AND-Verknüpfung abgeglichen.
  - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressourcen.
  - **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
  - **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
  - **resourceMatchers[].names**: (*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-Selektorzeichenfolge im Kubernetes-Metadatenfeld „name“ der Ressource, wie in der Dokumentation 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 `trident-protect-snapshot-restore-cr.yaml` Datei mit den korrekten Werten, CR anwenden:

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

Verwenden der CLI

Schritte

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

- Der `snapshot` Das Argument verwendet einen Namespace und einen Snapshot-Namen im folgenden Format: `<namespace>/<name>` .
- Der `namespace-mapping` Das Argument verwendet durch Doppelpunkte getrennte Namensräume, um Quell-Namensräume den korrekten Ziel-Namensräumen im folgenden Format 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 aus einem Snapshot in den ursprünglichen Namensraum

Sie können jederzeit einen Snapshot im ursprünglichen Namensraum wiederherstellen.

> Wenn Ihre Anwendung mehrere Namespaces verwendet und diese Namespaces PVCs mit demselben Namen haben, funktionieren Snapshot-Wiederherstellungsvorgänge (sowohl direkt als auch in einen neuen Namespace) nicht korrekt. Alle wiederhergestellten Volumes haben dieselben Daten anstatt der korrekten Daten für jeden Namespace. Verwenden Sie die Backup-Wiederherstellung anstelle der Snapshot-Wiederherstellung oder aktualisieren Sie auf Version 26.02 oder höher, die dieses Problem behebt.

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger andauernden S3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Siehe die "AWS API-Dokumentation" Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie hier.
- Siehe die "AWS IAM-Dokumentation" Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie hier.

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) und benennen Sie sie. `trident-protect-snapshot-ipr-cr.yaml` .
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und aussagekräftigen Namen für Ihre Umgebung.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVaults, in dem die Snapshot-Inhalte gespeichert sind.
   - **spec.appArchivePath**: Der Pfad innerhalb von AppVault, in dem die Snapshot-Inhalte gespeichert sind. Sie können 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*) Falls Sie nur bestimmte Ressourcen der Anwendung für die Wiederherstellung auswählen möchten, fügen Sie Filter hinzu, die Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließen:

   > 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**: (Für die Filterung erforderlich) Verwenden Sie `Include` oder `Exclude` Eine in resourceMatchers definierte Ressource ein- oder ausschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die ein- oder auszuschließenden Ressourcen zu definieren:
     - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese per OR-Verknüpfung abgeglichen, und die Felder innerhalb jedes Elements (Gruppe, Art, Version) werden per AND-Verknüpfung abgeglichen.
       - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressourcen.
       - **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.

- **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
- **resourceMatchers[].names**: (*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-Selektorzeichenfolge im Kubernetes-Metadatenfeld „name“ der Ressource, wie in der Dokumentation 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 `trident-protect-snapshot-ipr-cr.yaml` Datei mit den korrekten Werten, CR anwenden:

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

Verwenden der CLI

Schritte

1. Stellen Sie den Snapshot im ursprünglichen Namespace wieder her und ersetzen Sie die 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 läuft, abgeschlossen ist oder fehlgeschlagen ist.

Schritte

1. Verwenden Sie den folgenden Befehl, um den Status des Wiederherstellungsvorgangs abzurufen. Ersetzen Sie dabei die Werte in Klammern durch Informationen aus Ihrer Umgebung:

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

Verwenden Sie die erweiterten Trident Protect-Wiederherstellungseinstellungen.

Sie können Wiederherstellungsvorgänge mithilfe erweiterter Einstellungen wie Anmerkungen, Namespace-Einstellungen und Speicheroptionen an Ihre spezifischen Anforderungen anpassen.

Namespace-Annotationen und -Bezeichnungen während Wiederherstellungs- und Failover-Operationen

Bei Wiederherstellungs- und Failover-Vorgängen werden die Bezeichnungen und Annotationen im Ziel-Namensraum so angepasst, dass sie mit den Bezeichnungen und Annotationen im Quell-Namensraum übereinstimmen. Es werden Bezeichnungen oder Annotationen aus dem Quell-Namensraum hinzugefügt, die im Ziel-Namensraum nicht existieren, und alle bereits vorhandenen Bezeichnungen oder Annotationen werden überschrieben, um dem Wert aus dem Quell-Namensraum zu entsprechen. Labels oder Annotationen, die nur im Ziel-Namensraum existieren, bleiben unverändert.

> Wenn Sie Red Hat OpenShift verwenden, ist es wichtig, die entscheidende Rolle von Namespace-Annotationen in OpenShift-Umgebungen zu beachten. Namespace-Anmerkungen stellen sicher, dass wiederhergestellte Pods die entsprechenden Berechtigungen und Sicherheitskonfigurationen einhalten, die durch die OpenShift-Sicherheitskontextbeschränkungen (SCCs) definiert sind, und ohne Berechtigungsprobleme auf Volumes zugreifen können. Weitere Informationen finden Sie unter "Dokumentation zu den Sicherheitskontextbeschränkungen von OpenShift" .

Sie können verhindern, dass bestimmte Annotationen im Ziel-Namespace überschrieben werden, indem Sie die Kubernetes-Umgebungsvariable festlegen. `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` bevor Sie die Wiederherstellungs- oder Failover-Operation durchführen. Beispiel:

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

Bei der Durchführung einer Wiederherstellungs- oder Failover-Operation werden alle in angegebenen Namespace-Annotationen und -Labels berücksichtigt. `restoreSkipNamespaceAnnotations` Und `restoreSkipNamespaceLabels` sind von der Wiederherstellungs- oder Failover-Operation ausgeschlossen. Stellen Sie sicher, dass diese Einstellungen während der ersten Helm-Installation konfiguriert werden. Weitere Informationen finden Sie unter "Konfigurieren Sie AutoSupport und Namespace-Filteroptionen".

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

Das folgende Beispiel zeigt einen Quell- und einen Ziel-Namensraum, die jeweils unterschiedliche Annotationen und Bezeichnungen aufweisen. Sie können den Zustand des Ziel-Namensraums vor und nach der Operation einsehen und erkennen, wie die Annotationen und Labels im Ziel-Namensraum kombiniert oder überschrieben werden.

Vor dem Wiederherstellungs- oder Failover-Vorgang

Die folgende Tabelle veranschaulicht den Zustand der Beispiel-Quell- und Ziel-Namespaces vor der Wiederherstellungs- oder Failover-Operation:

Namensraum	Anmerkungen	Labels
Namespace ns-1 (Quelle)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• Umgebung=Produktion • Compliance=HIPAA • Name=ns-1
Namespace ns-2 (Ziel)	• annotation.one/key: "true" • annotation.three/key: "false"	• Rolle=Datenbank

Nach dem Wiederherstellungsvorgang

Die folgende Tabelle veranschaulicht den Zustand des Beispiel-Ziel-Namespace nach der Wiederherstellungs- oder Failover-Operation. Einige Schlüssel wurden hinzugefügt, einige wurden überschrieben, und die `name` Die Bezeichnung wurde aktualisiert, um dem Ziel-Namespace zu entsprechen:

Namensraum	Anmerkungen	Labels
Namespace ns-2 (Ziel)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• Name=ns-2 • Compliance=HIPAA • Umgebung=Produktion • Rolle=Datenbank

Unterstützte Felder

In diesem Abschnitt werden zusätzliche Felder beschrieben, die für Wiederherstellungsvorgänge verfügbar sind.

Speicherklassenzuordnung

Der `spec.storageClassMapping` Das Attribut definiert eine Zuordnung von einer in der Quellanwendung vorhandenen Speicherklasse zu einer neuen Speicherklasse im Zielcluster. Sie können dies verwenden, wenn Sie Anwendungen zwischen Clustern mit unterschiedlichen Speicherklassen migrieren oder wenn Sie das Speicher-Backend für BackupRestore-Vorgänge ändern.

Beispiel:

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

Unterstützte Annotationen

Dieser Abschnitt listet die unterstützten Annotationen zur Konfiguration verschiedener Verhaltensweisen im System auf. Wenn eine Annotation nicht explizit vom Benutzer festgelegt wird, verwendet das System den Standardwert.

Anmerkung	Typ	Beschreibung	Standardwert
protect.trident.netapp.io/data-mover-timeout-sec	Schnur	Die maximal zulässige Zeit (in Sekunden) für einen Stillstand des Datenübertragungsvorgangs.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	Schnur	Die maximale Größenbeschränkung (in Megabyte) für den Kopia-Inhaltscache.	"1000"

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 -Bezeichnungen während Wiederherstellungs- und Failover-Operationen

Bei Wiederherstellungs- und Failover-Vorgängen werden die Bezeichnungen und Annotationen im Ziel-

Namensraum so angepasst, dass sie mit den Bezeichnungen und Annotationen im Quell-Namensraum übereinstimmen. Es werden Bezeichnungen oder Annotationen aus dem Quell-Namensraum hinzugefügt, die im Ziel-Namensraum nicht existieren, und alle bereits vorhandenen Bezeichnungen oder Annotationen werden überschrieben, um dem Wert aus dem Quell-Namensraum zu entsprechen. Labels oder Annotationen, die nur im Ziel-Namensraum existieren, bleiben unverändert.

> Wenn Sie Red Hat OpenShift verwenden, ist es wichtig, die entscheidende Rolle von Namespace-Annotationen in OpenShift-Umgebungen zu beachten. Namespace-Anmerkungen stellen sicher, dass wiederhergestellte Pods die entsprechenden Berechtigungen und Sicherheitskonfigurationen einhalten, die durch die OpenShift-Sicherheitskontextbeschränkungen (SCCs) definiert sind, und ohne Berechtigungsprobleme auf Volumes zugreifen können. Weitere Informationen finden Sie unter "Dokumentation zu den Sicherheitskontextbeschränkungen von OpenShift" .

Sie können verhindern, dass bestimmte Annotationen im Ziel-Namespace überschrieben werden, indem Sie die Kubernetes-Umgebungsvariable festlegen. `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` bevor Sie die Wiederherstellungs- oder Failover-Operation durchführen. Beispiel:

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

> Bei der Durchführung einer Wiederherstellungs- oder Failover-Operation werden alle in angegebenen Namespace-Annotationen und -Labels berücksichtigt. `restoreSkipNamespaceAnnotations` Und `restoreSkipNamespaceLabels` sind von der Wiederherstellungs- oder Failover-Operation ausgeschlossen. Stellen Sie sicher, dass diese Einstellungen während der ersten Helm-Installation konfiguriert werden. Weitere Informationen finden Sie unter "Konfigurieren Sie AutoSupport und Namespace-Filteroptionen".

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

Das folgende Beispiel zeigt einen Quell- und einen Ziel-Namensraum, die jeweils unterschiedliche Annotationen und Bezeichnungen aufweisen. Sie können den Zustand des Ziel-Namensraums vor und nach der Operation einsehen und erkennen, wie die Annotationen und Labels im Ziel-Namensraum kombiniert oder überschrieben werden.

Vor dem Wiederherstellungs- oder Failover-Vorgang

Die folgende Tabelle veranschaulicht den Zustand der Beispiel-Quell- und Ziel-Namespaces vor der Wiederherstellungs- oder Failover-Operation:

Namensraum	Anmerkungen	Labels
Namespace ns-1 (Quelle)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• Umgebung=Produktion • Compliance=HIPAA • Name=ns-1
Namespace ns-2 (Ziel)	• annotation.one/key: "true" • annotation.three/key: "false"	• Rolle=Datenbank

Nach dem Wiederherstellungsvorgang

Die folgende Tabelle veranschaulicht den Zustand des Beispiel-Ziel-Namespace nach der Wiederherstellungs- oder Failover-Operation. Einige Schlüssel wurden hinzugefügt, einige wurden überschrieben, und die `name` Die Bezeichnung wurde aktualisiert, um dem Ziel-Namespace zu entsprechen:

Namensraum	Anmerkungen	Labels
Namespace ns-2 (Ziel)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "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.".

Ausführungs-Hooks während Failover- und Rückwärtsoperationen

Bei der Verwendung einer AppMirror-Beziehung zum Schutz Ihrer Anwendung gibt es spezifische Verhaltensweisen im Zusammenhang mit Ausführungs-Hooks, die Sie während Failover- und Rückwärtsoperationen beachten sollten.

- Beim Failover werden die Ausführungs-Hooks automatisch vom Quellcluster in den Zielcluster kopiert. Sie müssen sie nicht manuell neu erstellen. Nach einem Failover sind Ausführungs-Hooks in der Anwendung vorhanden und werden bei allen relevanten Aktionen ausgeführt.
- Bei einer umgekehrten Synchronisierung oder einer umgekehrten Resynchronisierung werden alle vorhandenen Ausführungs-Hooks der Anwendung entfernt. Wenn die Quellanwendung zur Zielanwendung wird, sind diese Ausführungs-Hooks ungültig und werden gelöscht, um ihre Ausführung zu verhindern.

Weitere Informationen zu Ausführungshooks finden Sie unter"Trident Protect-Ausführungshooks verwalten" .

Eine Replikationsbeziehung einrichten

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 Replikationszeitplans (einschließlich Kubernetes-Ressourcen sowie persistenter Volume-Daten)
- Einstellen des Zeitpunkts für die Aufnahme des Schnappschusses

Schritte

1. Erstellen Sie auf dem Quellcluster einen AppVault für die Quellanwendung. Je nach Speicheranbieter können Sie ein Beispiel in folgendem Format anpassen:"AppVault-Benutzerressourcen" um sich Ihrer Umgebung anzupassen:

Erstellen Sie einen AppVault mithilfe einer CR

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

b. Konfigurieren Sie die folgenden Attribute:

- **metadata.name**: (*Erforderlich*) Der Name der AppVault-Benutzerressource. Notieren Sie sich den gewählten Namen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diesen Wert verweisen.
- **spec.providerConfig**: (*Erforderlich*) Speichert die Konfiguration, die für den Zugriff auf den AppVault mit dem angegebenen Provider erforderlich ist. Wählen Sie einen Bucket-Namen und alle weiteren erforderlichen Details für Ihren Anbieter. Notieren Sie sich die von Ihnen gewählten Werte, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diese Werte verweisen. Siehe"AppVault-Benutzerressourcen" Beispiele für AppVault CRs bei anderen Anbietern finden Sie hier.
- **spec.providerCredentials**: (*Erforderlich*) Speichert Verweise auf alle Anmeldeinformationen, die für den Zugriff auf den AppVault mit dem angegebenen Provider erforderlich sind.
  - **spec.providerCredentials.valueFromSecret**: (*Erforderlich*) Gibt an, dass der Wert der Anmeldeinformationen aus einem Geheimnis stammen soll.
    - **Schlüssel**: (*Erforderlich*) Der gültige Schlüssel des Geheimnisses, aus dem ausgewählt werden soll.
    - **Name**: (*Erforderlich*) Name des Geheimnisses, das den Wert für dieses Feld enthält. Muss sich im selben Namensraum befinden.
  - **spec.providerCredentials.secretAccessKey**: (*Erforderlich*) Der Zugriffsschlüssel, der für den Zugriff auf den Provider verwendet wird. Der **Name** sollte mit **spec.providerCredentials.valueFromSecret.name** übereinstimmen.
- **spec.providerType**: (*Erforderlich*) Legt fest, was die Datensicherung bereitstellt; zum Beispiel NetApp ONTAP S3, generisches S3, Google Cloud oder Microsoft Azure. Mögliche Werte:
  - AWS
  - azurblau
  - gcp
  - generisches-s3
  - ontap-s3
  - Speichergitter-S3

c. Nachdem Sie die `trident-protect-appvault-primary-source.yaml` Datei mit den korrekten Werten, CR anwenden:

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

Erstellen Sie einen AppVault mithilfe der CLI.

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

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

2. Erstellen Sie auf dem Quellcluster die Quellanwendung CR:

Erstellen Sie die Quellanwendung mithilfe eines CR

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

b. Konfigurieren Sie die folgenden Attribute:

- **metadata.name**: (*Erforderlich*) Der Name der benutzerdefinierten Anwendungsressource. Notieren Sie sich den gewählten Namen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diesen Wert verweisen.
- **spec.includedNamespaces**: (*Erforderlich*) Ein Array von Namespaces und zugehörigen Bezeichnungen. Verwenden Sie Namespace-Namen und grenzen Sie optional den Geltungsbereich der Namespaces mit Bezeichnungen ein, um Ressourcen anzugeben, die in den hier aufgeführten Namespaces vorhanden sind. Der Anwendungs-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 `trident-protect-app-source.yaml` Datei mit den korrekten Werten, CR anwenden:

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

Erstellen Sie die Quellanwendung mithilfe der Befehlszeilenschnittstelle.

a. Erstellen Sie die Quellanwendung. Beispiel:

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

3. Optional kann auf dem Quellcluster ein Snapshot der Quellanwendung erstellt werden. Dieser Snapshot dient als Grundlage für die Anwendung auf dem Zielcluster. Wenn Sie diesen Schritt überspringen, müssen Sie warten, bis der nächste geplante Snapshot erstellt wird, damit Sie über einen aktuellen Snapshot verfügen. Informationen zum Erstellen eines On-Demand-Snapshots finden Sie unter "Erstellen Sie einen

On-Demand-Snapshot"Die

4. Erstellen Sie auf dem Quellcluster den Replikationsplan CR:

> Zusätzlich zum unten angegebenen Zeitplan wird empfohlen, einen separaten täglichen Snapshot-Zeitplan mit einer Aufbewahrungsdauer von 7 Tagen zu erstellen, um einen gemeinsamen Snapshot zwischen verbundenen ONTAP Clustern zu gewährleisten. Dadurch wird sichergestellt, dass Snapshots bis zu 7 Tage lang verfügbar sind, die Aufbewahrungsdauer kann jedoch an die Bedürfnisse des Benutzers angepasst werden.
>
> Im Falle eines Failovers kann das System diese Snapshots bis zu 7 Tage lang für Rückoperationen nutzen. Dieser Ansatz beschleunigt und optimiert den umgekehrten Prozess, da nur die seit dem letzten Snapshot vorgenommenen Änderungen übertragen werden, nicht aber alle Daten.
>
> Wenn ein bestehender Zeitplan für die Anwendung bereits die gewünschten Aufbewahrungsanforderungen erfüllt, sind keine zusätzlichen Zeitpläne erforderlich.

Erstellen Sie den Replikationszeitplan mithilfe einer CR.

a. Erstellen Sie einen Replikationszeitplan für die Quellanwendung:

i. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) und benennen Sie sie (zum Beispiel). `trident-protect-schedule.yaml`).

ii. Konfigurieren Sie die folgenden Attribute:

- **metadata.name**: (*Erforderlich*) Der Name der benutzerdefinierten Zeitplanressource.
- **spec.appVaultRef**: (*Erforderlich*) Dieser Wert muss mit dem Feld metadata.name des AppVaults für die Quellanwendung übereinstimmen.
- **spec.applicationRef**: (*Erforderlich*) Dieser Wert muss mit dem Feld metadata.name der Quellanwendung CR übereinstimmen.
- **spec.backupRetention**: (*Erforderlich*) Dieses Feld ist erforderlich und muss den Wert 0 haben.
- **spec.enabled**: Muss auf true gesetzt sein.
- **spec.granularity**: Muss auf gesetzt werden `Custom` .
- **spec.recurrenceRule**: Definiert 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
  namespace: my-app-namespace
spec:
  appVaultRef: my-appvault-name
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: Custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

i. Nachdem Sie die `trident-protect-schedule.yaml` Datei mit den korrekten Werten, CR anwenden:

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

Erstellen Sie den Replikationszeitplan mithilfe der Befehlszeilenschnittstelle.

a. Erstellen Sie den Replikationszeitplan und ersetzen Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung:

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

Beispiel:

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

5. Erstellen Sie auf dem Zielcluster eine AppVault-CR für die Quellanwendung, die mit der auf dem Quellcluster angewendeten AppVault-CR identisch ist, und benennen Sie sie (zum Beispiel). `trident-protect-appvault-primary-destination.yaml`).
6. Wenden Sie die CR an:

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

7. Erstellen Sie einen AppVault CR für die Zielanwendung auf dem Zielcluster. Je nach Speicheranbieter können Sie ein Beispiel in folgendem Format anpassen:"AppVault-Benutzerressourcen" um sich Ihrer Umgebung anzupassen:
   a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) und benennen Sie sie (zum Beispiel). `trident-protect-appvault-secondary-destination.yaml`).
   b. Konfigurieren Sie die folgenden Attribute:
      - **metadata.name**: (*Erforderlich*) Der Name der AppVault-Benutzerressource. Notieren Sie sich den gewählten Namen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diesen Wert verweisen.
      - **spec.providerConfig**: (*Erforderlich*) Speichert die Konfiguration, die für den Zugriff auf den AppVault mit dem angegebenen Provider erforderlich ist. Wählen Sie eine `bucketName` und alle weiteren für Ihren Anbieter notwendigen Angaben. Notieren Sie sich die von Ihnen gewählten Werte, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diese Werte verweisen.

Siehe"AppVault-Benutzerressourcen" Beispiele für AppVault CRs bei anderen Anbietern finden Sie hier.

- **spec.providerCredentials**: (*Erforderlich*) Speichert Verweise auf alle Anmeldeinformationen, die für den Zugriff auf den AppVault mit dem angegebenen Provider erforderlich sind.
  - **spec.providerCredentials.valueFromSecret**: (*Erforderlich*) Gibt an, dass der Wert der Anmeldeinformationen aus einem Geheimnis stammen soll.
    - **Schlüssel**: (*Erforderlich*) Der gültige Schlüssel des Geheimnisses, aus dem ausgewählt werden soll.
    - **Name**: (*Erforderlich*) Name des Geheimnisses, das den Wert für dieses Feld enthält. Muss sich im selben Namensraum befinden.
  - **spec.providerCredentials.secretAccessKey**: (*Erforderlich*) Der Zugriffsschlüssel, der für den Zugriff auf den Provider verwendet wird. Der **Name** sollte mit **spec.providerCredentials.valueFromSecret.name** übereinstimmen.
- **spec.providerType**: (*Erforderlich*) Legt fest, was die Datensicherung bereitstellt; zum Beispiel NetApp ONTAP S3, generisches S3, Google Cloud oder Microsoft Azure. Mögliche Werte:
  - AWS
  - azurblau
  - gcp
  - generisches-s3
  - ontap-s3
  - Speichergitter-S3

c. Nachdem Sie die `trident-protect-appvault-secondary-destination.yaml` Datei mit den korrekten Werten, CR anwenden:

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

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

Erstellen Sie eine AppMirrorRelationship mithilfe eines CR

a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) und benennen Sie sie (zum Beispiel). `trident-protect-relationship.yaml`).

b. Konfigurieren Sie die folgenden Attribute:

- **metadata.name:** (Erforderlich) Der Name der benutzerdefinierten Ressource AppMirrorRelationship.
- **spec.destinationAppVaultRef**: (*Erforderlich*) Dieser Wert muss mit dem Namen des AppVaults für die Zielanwendung auf dem Zielcluster übereinstimmen.
- **spec.namespaceMapping**: (*Erforderlich*) Die Ziel- und Quell-Namespaces müssen mit dem in der jeweiligen Anwendungs-CR definierten Anwendungs-Namespace übereinstimmen.
- **spec.sourceAppVaultRef**: (*Erforderlich*) Dieser Wert muss mit dem Namen des AppVaults für die Quellanwendung übereinstimmen.
- **spec.sourceApplicationName**: (*Erforderlich*) Dieser Wert muss mit dem Namen der Quellanwendung übereinstimmen, die Sie im Quellanwendungs-CR definiert haben.
- **spec.sourceApplicationUID**: (Erforderlich) Dieser Wert muss mit der UID der Quellanwendung übereinstimmen, die Sie im Quellanwendungs-CR definiert haben.
- **spec.storageClassName**: (*Optional*) Wählen Sie den Namen einer gültigen Speicherklasse im Cluster. Die Speicherklasse muss mit einer ONTAP Speicher-VM verknüpft sein, die mit der Quellumgebung per Peering verbunden ist. Wird keine Speicherklasse angegeben, wird standardmäßig die im Cluster vorhandene Standard-Speicherklasse verwendet.
- **spec.recurrenceRule**: Definiert 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 `trident-protect-relationship.yaml` Datei mit den korrekten Werten, CR anwenden:

```
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 die Werte in Klammern durch Informationen aus Ihrer Umgebung:

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

Beispiel:

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

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

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

Failover zum Zielcluster

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 auf dem Zielcluster die AppMirrorRelationship-CR-Datei (zum Beispiel, `trident-protect-relationship.yaml`) und ändern Sie den Wert von **spec.desiredState** in `Promoted` .
2. Speichern Sie die CR-Datei.
3. Wenden Sie die CR an:

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

4. (*Optional*) Erstellen Sie alle benötigten Schutzzeitpläne für die ausgefallene Anwendung.
5. (*Optional*) Überprüfen Sie den Status und die Stellung der Replikationsbeziehung:

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

Eine fehlgeschlagene Replikationsbeziehung erneut synchronisieren

Der Resynchronisierungsvorgang stellt die Replikationsbeziehung wieder her. Nach der Durchführung einer Resynchronisierung wird die ursprüngliche Quellanwendung zur laufenden Anwendung, und alle Änderungen, die an der laufenden Anwendung auf dem Zielcluster vorgenommen wurden, werden verworfen.

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

Alle Daten, die während des Failovers in die Zielanwendung geschrieben werden, gehen verloren.

Schritte

1. Optional: Erstellen Sie auf dem Quellcluster einen Snapshot der Quellanwendung. Dadurch wird sichergestellt, dass die neuesten Änderungen aus dem Quellcluster erfasst werden.
2. Bearbeiten Sie auf dem Zielcluster die AppMirrorRelationship-CR-Datei (zum Beispiel, `trident-protect-relationship.yaml`) und ändern Sie den Wert von spec.desiredState auf `Established` .
3. Speichern Sie die CR-Datei.
4. Wenden Sie die CR an:

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

5. Falls Sie auf dem Zielcluster Schutzpläne erstellt haben, um die ausgefallene Anwendung zu schützen, entfernen Sie diese. Alle verbleibenden Zeitpläne führen zu Fehlern bei der Erstellung von Volume-Snapshots.

Umgekehrte Resynchronisierung einer fehlgeschlagenen Replikationsbeziehung

Wenn Sie eine Replikationsbeziehung, die aufgrund eines Fehlers ausgefallen ist, rückgängig machen, wird die Zielanwendung zur Quellanwendung und die Quelle zum Ziel. Änderungen, die während des Failovers an der Zielanwendung vorgenommen werden, bleiben erhalten.

Schritte

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

Replikationsrichtung der Anwendung 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 werden Quelle und Ziel vertauscht.

Schritte

1. Erstellen Sie auf dem Quellcluster einen Shutdown-Snapshot:

Erstellen Sie einen Shutdown-Snapshot mithilfe eines CR

a. Deaktivieren Sie die Zeitpläne der Schutzrichtlinien für die Quellanwendung.

b. Erstellen Sie eine ShutdownSnapshot-CR-Datei:

   i. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) und benennen Sie sie (zum Beispiel). `trident-protect-shutdownsnapshot.yaml`).

   ii. Konfigurieren Sie die folgenden Attribute:

   - **metadata.name**: (*Erforderlich*) Der Name der benutzerdefinierten Ressource.
   - **spec.AppVaultRef**: (*Erforderlich*) Dieser Wert muss mit dem Feld metadata.name des AppVaults für die Quellanwendung übereinstimmen.
   - **spec.ApplicationRef**: (*Erforderlich*) Dieser Wert muss mit dem Feld metadata.name der Quell-CR-Datei der Anwendung 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 `trident-protect-shutdownsnapshot.yaml` Datei mit den korrekten Werten, CR anwenden:

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

Erstellen Sie einen Shutdown-Snapshot mithilfe der CLI.

a. Erstellen Sie den Shutdown-Snapshot und ersetzen Sie die 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 Shutdown-Snapshots den Status des Shutdown-Snapshots ab:

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

3. Ermitteln Sie auf dem 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 ein Failover vom neuen Zielcluster zum neuen Quellcluster mit folgender Änderung durch:

> Fügen Sie im zweiten Schritt des Failover-Verfahrens Folgendes hinzu: `spec.promotedSnapshot` Setzen Sie den Wert des Feldes in der AppMirrorRelationship CR-Datei auf den Basisnamen, den Sie in Schritt 3 oben aufgezeichnet haben.

5. Führen Sie die Schritte zur umgekehrten Resynchronisierung durch inUmgekehrte Resynchronisierung einer fehlgeschlagenen Replikationsbeziehung .
6. Aktivieren Sie die Schutzzeitpläne auf dem neuen Quellcluster.

Ergebnis

Die folgenden Aktionen erfolgen aufgrund der umgekehrten Replikation:

- Es wird eine Momentaufnahme der Kubernetes-Ressourcen der ursprünglichen Quellanwendung erstellt.
- Die Pods der ursprünglichen Quell-App werden ordnungsgemäß gestoppt, indem die Kubernetes-Ressourcen der App gelöscht werden (PVCs und PVs bleiben erhalten).
- Nach dem Herunterfahren der Pods werden Snapshots der App-Volumes erstellt und repliziert.
- Die SnapMirror -Beziehungen wurden aufgehoben, wodurch die Zielvolumes zum Lesen/Schreiben bereit sind.
- Die Kubernetes-Ressourcen der App werden aus dem Snapshot vor dem Herunterfahren wiederhergestellt, wobei die Volume-Daten verwendet werden, die nach dem Herunterfahren der ursprünglichen Quell-App repliziert wurden.
- Die Replikation wird in umgekehrter Richtung wiederhergestellt.

Failback-Anwendungen auf den ursprünglichen Quellcluster

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, die einen Failover auf ein Ziel abgeschlossen hat, und umfasst die folgenden Schritte:

- Beginnen Sie mit einem Ausfallzustand.
- Die Replikationsbeziehung wird umgekehrt neu synchronisiert.

> Führen Sie keine normale Resynchronisierungsoperation durch, da dadurch die während des Failover-Vorgangs auf den Zielcluster geschriebenen Daten verworfen werden.

- Die Replikationsrichtung umkehren.

Schritte

1. Führe dieUmgekehrte Resynchronisierung einer fehlgeschlagenen Replikationsbeziehung Schritte.
2. Führe dieReplikationsrichtung der Anwendung umkehren Schritte.

Eine Replikationsbeziehung löschen

Sie können eine Replikationsbeziehung jederzeit löschen. Wenn Sie die Replikationsbeziehung der Anwendung löschen, entstehen zwei separate Anwendungen ohne jegliche Beziehung zueinander.

Schritte

1. Löschen Sie im aktuellen Zielcluster die AppMirrorRelationship-CR:

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

Anwendungen mit Trident Protect migrieren

Sie können Ihre Anwendungen zwischen Clustern oder in verschiedene Speicherklassen migrieren, indem Sie Sicherungsdaten wiederherstellen.

> Bei der Migration einer Anwendung werden alle für die Anwendung konfigurierten Ausführungs-Hooks mit der Anwendung migriert. Wenn ein Hook für die Nachbearbeitung der Wiederherstellung vorhanden ist, wird dieser automatisch im Rahmen des Wiederherstellungsvorgangs ausgeführt.

Sicherungs- und Wiederherstellungsvorgänge

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

Klonen auf denselben Cluster

Um eine Anwendung auf denselben Cluster zu klonen, erstellen Sie einen Snapshot oder eine Sicherung und stellen Sie die Daten auf demselben Cluster wieder her.

Schritte

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

2. Führen Sie auf demselben Cluster einen der folgenden Schritte aus, je nachdem, ob Sie einen Snapshot oder eine Sicherung erstellt haben:
   a. "Stellen Sie Ihre Daten aus dem Snapshot wieder her.".
   b. "Stellen Sie Ihre Daten aus der Sicherung wieder her.".

Klonen auf anderen 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 replizieren, indem Sie"SnapMirror -Replikation" .

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 der Sicherung wieder her." .

Anwendungen von einer Speicherklasse in eine andere Speicherklasse migrieren

Sie können Anwendungen von einer Speicherklasse in eine andere Speicherklasse migrieren, indem Sie eine Sicherung in der Zielspeicherklasse wiederherstellen.

Zum 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 einer CR wieder her.

Schritte

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

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

   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVaults, in dem die Snapshot-Inhalte gespeichert sind.
   - **spec.namespaceMapping**: Die Zuordnung des Quell-Namespace der Wiederherstellungsoperation zum Ziel-Namespace. Ersetzen `my-source-namespace` 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. Optional können Sie, falls Sie nur bestimmte Ressourcen der Anwendung für die Wiederherstellung auswählen möchten, Filter hinzufügen, die Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließen:
   - **resourceFilter.resourceSelectionCriteria**: (Für die Filterung erforderlich) Verwenden Sie `include or exclude` Eine in resourceMatchers definierte Ressource ein- oder ausschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die ein- oder auszuschließenden Ressourcen zu definieren:
     - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese per OR-Verknüpfung abgeglichen, und die Felder innerhalb jedes Elements (Gruppe, Art, Version) werden per AND-Verknüpfung abgeglichen.
       - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressourcen.

- **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
- **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
- **resourceMatchers[].names**: (*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-Selektorzeichenfolge im Kubernetes-Metadatenfeld „name“ der Ressource, wie in der Dokumentation 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 `trident-protect-snapshot-restore-cr.yaml` Datei mit den korrekten Werten, CR anwenden:

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

Stellen Sie den Snapshot mithilfe der Befehlszeile wieder her.

Schritte

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

   - Der `snapshot` Das Argument verwendet einen Namespace und einen Snapshot-Namen im folgenden Format: `<namespace>/<name>` .
   - Der `namespace-mapping` Das Argument verwendet durch Doppelpunkte getrennte Namensräume, um Quell-Namensräume den korrekten Ziel-Namensräumen im folgenden Format 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 Ausführungshook ist eine benutzerdefinierte Aktion, die Sie so konfigurieren können, dass sie in Verbindung mit einer Datensicherungsoperation einer verwalteten App ausgeführt wird. Wenn Sie beispielsweise über eine Datenbank-App verfügen, können Sie mithilfe eines Ausführungs-Hooks alle Datenbanktransaktionen vor einem Snapshot anhalten und die Transaktionen nach Abschluss des Snapshots fortsetzen. Dadurch werden anwendungskonsistente Snapshots gewährleistet.

Arten von Ausführungs-Hooks

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

- Vorab-Schnappschuss
- Nach dem Snapshot
- Vorsicherung
- Nach der Sicherung
- Nach der Wiederherstellung
- Nach dem Failover

Reihenfolge der Ausführung

Wenn ein Datenschutzvorgang ausgeführt wird, finden Ausführungs-Hook-Ereignisse in der folgenden Reihenfolge statt:

1. Alle anwendbaren benutzerdefinierten Ausführungs-Hooks vor der Operation werden auf den entsprechenden Containern ausgeführt. Sie können so viele benutzerdefinierte Hooks vor der Operation erstellen und ausführen, wie Sie benötigen, aber die Reihenfolge der Ausführung dieser Hooks 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 Datenschutzvorgang wird durchgeführt.
4. Eingefrorene Dateisysteme werden gegebenenfalls wieder freigegeben.
5. Alle anwendbaren benutzerdefinierten Ausführungs-Hooks nach der Operation werden auf den entsprechenden Containern ausgeführt. Sie können so viele benutzerdefinierte Post-Operation-Hooks erstellen und ausführen, wie Sie benötigen, aber die Reihenfolge der Ausführung dieser Hooks nach der Operation ist weder garantiert noch konfigurierbar.

Wenn Sie mehrere Ausführungs-Hooks desselben Typs erstellen (z. B. Pre-Snapshot), ist die Ausführungsreihenfolge dieser Hooks nicht garantiert. Die Reihenfolge der Ausführung von Hooks unterschiedlichen Typs ist jedoch gewährleistet. Im Folgenden sehen Sie beispielsweise die Ausführungsreihenfolge einer Konfiguration, die alle verschiedenen Hook-Typen enthält:

1. Vor dem Snapshot ausgeführte Hooks
2. Nach dem Snapshot ausgeführte Hooks
3. Vor der Sicherung ausgeführte Hooks
4. Nach der Sicherung ausgeführte Hooks

Das vorhergehende Befehlsbeispiel gilt nur, wenn Sie eine Sicherung durchführen, die keinen vorhandenen Snapshot verwendet.

Sie sollten Ihre Ausführungs-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 Ausführungs-Hooks in einer Produktionsumgebung aktiviert haben, testen Sie die resultierenden Snapshots und Backups, um sicherzustellen, dass sie konsistent sind. Sie können dies tun, indem Sie die App in einen temporären Namespace klonen, den Snapshot oder das Backup wiederherstellen und dann die App testen.

Wenn ein Pre-Snapshot-Ausführungs-Hook Kubernetes-Ressourcen hinzufügt, ändert oder entfernt, werden diese Änderungen in den Snapshot oder die Sicherung und in alle nachfolgenden Wiederherstellungsvorgänge einbezogen.

Wichtige Hinweise zu benutzerdefinierten Ausführungs-Hooks

Berücksichtigen Sie Folgendes, wenn Sie Ausführungs-Hooks für Ihre Apps planen.

- Ein Ausführungs-Hook muss ein Skript verwenden, um Aktionen auszuführen. Viele Ausführungs-Hooks 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 Ausführungs-Hooks häufig die Funktionalität der Anwendung, für die sie ausgeführt werden, einschränken oder vollständig deaktivieren, sollten Sie immer versuchen, die Ausführungszeit Ihrer benutzerdefinierten Ausführungs-Hooks zu minimieren. Wenn Sie einen Sicherungs- oder Snapshot-Vorgang mit zugehörigen Ausführungs-Hooks starten, ihn dann aber abbrechen, können die Hooks weiterhin ausgeführt werden, wenn der Sicherungs- oder Snapshot-Vorgang bereits begonnen hat. Dies bedeutet, dass die in einem Ausführungs-Hook nach der Sicherung verwendete Logik nicht davon ausgehen kann, dass die Sicherung abgeschlossen wurde.

Ausführungs-Hook-Filter

Wenn Sie einen Ausführungs-Hook für eine Anwendung hinzufügen oder bearbeiten, können Sie dem Ausführungs-Hook Filter hinzufügen, um zu verwalten, mit welchen Containern der Hook übereinstimmt. Filter

sind nützlich für Anwendungen, die auf allen Containern dasselbe Container-Image verwenden, aber jedes Image möglicherweise für einen anderen Zweck verwenden (z. B. Elasticsearch). Mithilfe von Filtern können Sie Szenarien erstellen, in denen Ausführungs-Hooks auf einigen, aber nicht unbedingt allen identischen Containern ausgeführt werden. Wenn Sie mehrere Filter für einen einzelnen Ausführungs-Hook erstellen, werden diese mit einem logischen UND-Operator kombiniert. Sie können bis zu 10 aktive Filter pro Ausführungs-Hook 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 "Unterstützung der Syntax „Regulärer Ausdruck 2“ (RE2)" Die

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

Beispiele für Ausführungs-Hooks

Besuchen Sie die "NetApp Verda GitHub-Projekt" um echte Ausführungs-Hooks für beliebte Apps wie Apache Cassandra und Elasticsearch herunterzuladen. Sie können sich auch Beispiele ansehen und Ideen für die Strukturierung Ihrer eigenen benutzerdefinierten Ausführungs-Hooks holen.

Erstelle einen Ausführungs-Hook

Sie können einen benutzerdefinierten Ausführungshook für eine App erstellen, indem Sie . Sie benötigen die Berechtigungen Eigentümer, Administrator oder Mitglied, um Ausführungs-Hooks zu erstellen.

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) 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**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und aussagekräftigen Namen für Ihre Umgebung.
   - **spec.applicationRef**: (*Erforderlich*) Der Kubernetes-Name der Anwendung, für die der Ausführungshook ausgeführt werden soll.
   - **spec.stage**: (*Erforderlich*) Eine Zeichenkette, die angibt, in welcher Phase der Aktion der Ausführungs-Hook ausgeführt werden soll. Mögliche Werte:
     - Vor
     - Post
   - **spec.action**: (*Erforderlich*) Eine Zeichenkette, die angibt, welche Aktion der Ausführungshook ausführen wird, vorausgesetzt, alle angegebenen Ausführungshook-Filter stimmen überein. Mögliche Werte:
     - Schnappschuss
     - Sicherung
     - Wiederherstellen
     - Ausfallsicherung
   - **spec.enabled**: (*Optional*) Gibt an, ob dieser Ausführungs-Hook aktiviert oder deaktiviert ist. Wenn kein Wert angegeben ist, ist der Standardwert „true“.
   - **spec.hookSource**: (*Erforderlich*) Eine Zeichenkette, die das Base64-kodierte Hook-Skript enthält.
   - **spec.timeout**: (*Optional*) Eine Zahl, die angibt, wie lange in Minuten der Ausführungs-Hook ausgeführt werden darf. Der Mindestwert beträgt 1 Minute, der Standardwert beträgt 25 Minuten, sofern nichts anderes angegeben ist.
   - **spec.arguments**: (*Optional*) Eine YAML-Liste von Argumenten, die Sie für den Ausführungs-Hook angeben können.
   - **spec.matchingCriteria**: (*Optional*) Eine optionale Liste von Kriterien-Schlüssel-Wert-Paaren, wobei jedes Paar einen Ausführungs-Hook-Filter bildet. Sie können pro Ausführungs-Hook bis zu 10 Filter hinzufügen.
   - **spec.matchingCriteria.type**: (*Optional*) Eine Zeichenkette, die den Typ des Ausführungs-Hook-Filters identifiziert. Mögliche Werte:
     - ContainerImage
     - ContainerName
     - PodName
     - PodLabel
     - NamespaceName
   - **spec.matchingCriteria.value**: (*Optional*) Eine Zeichenkette oder ein regulärer Ausdruck, der den Wert des Ausführungs-Hook-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 korrekten Werten gefüllt haben, wenden Sie die CR an:

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

Verwenden der CLI

Schritte

1. Erstellen Sie den Ausführungs-Hook und ersetzen Sie die 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>
```

Einen Ausführungs-Hook manuell ausführen

Sie können einen Ausführungshook manuell ausführen, beispielsweise zu Testzwecken oder wenn Sie den Hook nach einem Fehler manuell erneut ausführen müssen. Sie benötigen Eigentümer-, Administrator- oder Mitgliedsberechtigungen, um Ausführungshooks manuell auszuführen.

Das manuelle Ausführen eines Ausführungshooks besteht aus zwei grundlegenden Schritten:

1. Erstellen Sie eine Ressourcensicherung, die Ressourcen sammelt und eine Sicherungskopie davon erstellt, um festzulegen, wo der Hook ausgeführt wird.
2. Führe den Ausführungshook für das Backup aus.

Schritt 1: Erstellen Sie eine Ressourcensicherung.

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) 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**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und aussagekräftigen Namen für Ihre Umgebung.
   - **spec.applicationRef**: (*Erforderlich*) Der Kubernetes-Name der Anwendung, für die das Ressourcen-Backup erstellt werden soll.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVaults, in dem die Sicherungsinhalte gespeichert sind.
   - **spec.appArchivePath**: Der Pfad innerhalb von AppVault, in dem die Sicherungsinhalte gespeichert sind. Sie können 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 korrekten Werten gefüllt haben, wenden Sie die CR an:

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

Verwenden der CLI

Schritte

1. Erstellen Sie die Sicherungskopie und ersetzen Sie die 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 der Datensicherung anzeigen. Sie können diesen Beispielbefehl so oft verwenden, bis der Vorgang abgeschlossen ist:

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

3. Überprüfen Sie, ob die Datensicherung erfolgreich war:

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

Schritt 2: Den Ausführungshook ausführen

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) 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**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und aussagekräftigen Namen für Ihre Umgebung.
   - **spec.applicationRef**: (*Erforderlich*) Stellen Sie sicher, dass dieser Wert mit dem Anwendungsnamen aus dem ResourceBackup CR übereinstimmt, den Sie in Schritt 1 erstellt haben.
   - **spec.appVaultRef**: (*Erforderlich*) Stellen Sie sicher, dass dieser Wert mit dem appVaultRef aus dem ResourceBackup CR übereinstimmt, den Sie in Schritt 1 erstellt haben.
   - **spec.appArchivePath**: Stellen Sie sicher, dass dieser Wert mit dem appArchivePath des 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**: (*Erforderlich*) Eine Zeichenkette, die angibt, welche Aktion der Ausführungshook ausführen wird, vorausgesetzt, alle angegebenen Ausführungshook-Filter stimmen überein. Mögliche Werte:
     - Schnappschuss
     - Sicherung
     - Wiederherstellen
     - Ausfallsicherung
   - **spec.stage**: (*Erforderlich*) Eine Zeichenkette, die angibt, in welcher Phase der Aktion der Ausführungs-Hook ausgeführt werden soll. Dieser Hakenlauf wird in keiner anderen Phase Haken ausführen. 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 korrekten Werten gefüllt haben, wenden Sie die CR an:

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

Verwenden der CLI

Schritte

1. Erstellen Sie die manuelle Ausführungs-Hook-Anforderung:

```
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-Hooks. Sie können diesen Befehl so oft ausführen, bis der Vorgang abgeschlossen ist:

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

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

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