



Anwendungen verwalten und schützen

Trident

NetApp
October 29, 2026

Inhalt

Anwendungen verwalten und schützen	1
Verwenden Sie Trident Protect AppVault-Objekte, um Buckets zu verwalten	1
Konfigurieren Sie die AppVault-Authentifizierung und Passwörter	1
AppVault-Erstellungsbeispiele	5
Informationen zu AppVault anzeigen	10
Entfernen Sie ein AppVault	11
Definieren Sie eine Anwendung für das Management mit Trident Protect	12
Erstellen Sie eine AppVault CR	12
Eine Anwendung definieren	12
Schützen Sie Anwendungen mit Trident Protect	17
Erstellen Sie einen On-Demand-Snapshot	17
Erstellen Sie eine bedarfsgesteuerte Datensicherung	19
Erstellen Sie einen Zeitplan für die Datensicherung	21
Einen Snapshot löschen	25
Ein Backup löschen	26
Überprüfen Sie den Status eines Sicherungsvorgangs	26
Backup und Restore für azure-netapp-files (ANF)-Vorgänge aktivieren	26
Anwendungen wiederherstellen	27
Anwendungen mit Trident Protect wiederherstellen	27
Verwenden Sie die erweiterten Trident Protect-Wiederherstellungseinstellungen	42
Ändern Sie Ressourcen während der Wiederherstellung mithilfe benutzerdefinierter Ressourcen	44
Anwendungen mit NetApp SnapMirror und Trident Protect replizieren	49
Namespace-Annotationen und -Labels während Wiederherstellungs- und Failover-Operationen	50
Ausführungshooks während Failover- und Reverse-Operationen	51
Eine Replikationsbeziehung einrichten	51
Replikationsrichtung der Anwendung umkehren	61
Anwendungen mit Trident Protect migrieren	63
Sicherungs- und Wiederherstellungsvorgänge	63
Anwendungen von einer Speicherklasse zu einer anderen Speicherklasse migrieren	64
Schutz von KubeVirt virtuellen Maschinen mit Trident Protect	67
KubeVirt Schutzmodi	67
Während der Sicherung und des Snapshots erfasste Ressourcen	68
VMs mit namespace-basierten Anwendungsdefinitionen schützen	69
Dynamischer VM-Schutz mit Label-Selektoren	71
Bestimmte VMs mit includedVirtualMachines schützen	72
Bestimmte KubeVirt VMs wiederherstellen	73
Eine VM-bezogene Anwendung mithilfe der Befehlszeilenschnittstelle erstellen	78
Verwandte Informationen	78
Trident Protect-Ausführungshooks verwalten	79
Arten von Execution Hooks	79
Wichtige Hinweise zu benutzerdefinierten Ausführungshooks	80
Ausführungs-Hook-Filter	80
Beispiele für Execution Hooks	81

Erstellen Sie einen Ausführungs-Hook	81
Einen Ausführungshook manuell ausführen	83

Anwendungen verwalten und schützen

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

Die Bucket-Custom-Resource (CR) für Trident Protect ist als AppVault bekannt. AppVault-Objekte sind die deklarative Kubernetes-Workflow-Darstellung eines Speicher-Buckets. Ein AppVault CR enthält die Konfigurationen, die erforderlich sind, damit ein Bucket in Datensicherungsoperationen wie Backups, Snapshots, Wiederherstellungsvorgängen und SnapMirror-Replikation verwendet werden kann. Nur Administratoren können AppVaults erstellen.

Sie müssen ein AppVault CR manuell oder über die Befehlszeile erstellen, wenn Sie Datensicherungsoperationen an einer Anwendung durchführen. Das AppVault CR ist spezifisch für Ihre Umgebung, und Sie können die Beispiele auf dieser Seite als Leitfaden verwenden, wenn Sie AppVault CRs erstellen.



Stellen Sie sicher, dass die AppVault CR auf dem Cluster vorhanden ist, auf dem Trident Protect installiert ist. Wenn die AppVault CR nicht existiert oder Sie nicht darauf zugreifen können, zeigt die Befehlszeile einen Fehler an.

Konfigurieren Sie die AppVault-Authentifizierung und Passwörter

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

Passwörter für das Data mover-Repository

Wenn Sie AppVault-Objekte mit 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 Secret angeben, verwendet Trident Protect ein Standardpasswort.

- Wenn Sie AppVault-CRs manuell erstellen, verwenden Sie das Feld **spec.dataMoverPasswordSecretRef**, um das Secret anzugeben.
- Beim Erstellen von AppVault-Objekten mit der Trident Protect CLI verwenden Sie das `--data-mover-password-secret-ref` Argument, um das Geheimnis anzugeben.

Erstellen Sie ein Passwortgeheimnis für das Data Mover Repository

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



- Je nachdem, welchen Datenmover Sie verwenden, müssen Sie nur das entsprechende Passwort für diesen Datenmover 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 sicher auf. Sie benötigen es, um Daten auf demselben Cluster oder auf einem anderen Cluster wiederherzustellen. Wenn der Cluster oder das `trident-`

protect 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 Sie die Befehlszeile

```
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, "[StorageGrid S3](#)", oder "[ONTAP S3](#)" mit Trident Protect zugreifen, müssen Sie sicherstellen, dass die von Ihnen bereitgestellten Benutzeranmeldeinformationen über die erforderlichen Berechtigungen für den Zugriff auf den Bucket verfügen. Im Folgenden finden Sie 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 die S3-kompatiblen Bucket-Richtlinien verwaltet.

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

```
]
}
```

Weitere Informationen zu Amazon S3-Richtlinien finden Sie in den Beispielen im ["Amazon S3-Dokumentation"](#).

EKS Pod Identity für Amazon S3 (AWS) Authentifizierung

Trident Protect unterstützt EKS Pod Identity für Kopia data mover-Operationen. Diese Funktion ermöglicht sicheren Zugriff auf S3-Buckets, ohne AWS-Anmeldeinformationen in Kubernetes-Secrets speichern zu müssen.

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 sind unter ["S3-kompatible Speicher-IAM-Berechtigungen"](#) zu finden.
- Die IAM-Rolle ist den folgenden Trident Protect service accounts 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 ["AWS EKS Pod Identity-Dokumentation"](#).

AppVault-Konfiguration Bei Verwendung von EKS Pod Identity konfigurieren Sie Ihre AppVault-CR mit dem `useIAM: true`-Flag anstelle 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

Beim Definieren eines AppVault CR müssen Sie Anmeldeinformationen angeben, um auf die vom Anbieter

gehosteten Ressourcen zuzugreifen, es sei denn, Sie verwenden IAM-Authentifizierung. Wie Sie die Schlüssel für die Anmeldeinformationen generieren, hängt vom jeweiligen Anbieter ab. Im Folgenden finden Sie Beispiele für die Schlüsselgenerierung über die Befehlszeile für verschiedene 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
```

AppVault-Erstellungsbeispiele

Die folgenden sind Beispiel-AppVault-Definitionen 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-Repositories enthält. Weitere Informationen finden Sie unter [Passwörter für das Data mover-Repository](#).
- Für Amazon S3 (AWS) AppVault-Objekte 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 in [AppVault Schlüsselgenerierungsbeispiele für Cloud-Anbieter](#) generieren.
- Für S3 AppVault-Objekte können Sie optional eine Egress-Proxy-URL für ausgehenden S3-Datenverkehr mithilfe des spec.providerConfig.S3.proxyURL Schlüssels angeben.

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
```



Für EKS-Umgebungen, die Pod Identity mit Kopia Data Mover verwenden, können Sie den `providerCredentials` Abschnitt entfernen und `useIAM: true` unter der `s3` Konfiguration hinzufügen.

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

```

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

```

AppVault Erstellungsbeispiele 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-Repositories enthält. Weitere Informationen finden Sie unter [Passwörter für das Data mover-Repository](#).

- Für S3-AppVault-Objekte können Sie optional eine Egress-Proxy-URL für ausgehenden S3-Datenverkehr mithilfe des `--proxy-url <ip_address:port>` Arguments angeben.

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
```

Unterstützte providerConfig.s3 Konfigurationsoptionen

Siehe die folgende Tabelle für die S3-Provider-Konfigurationsoptionen:

Parameter	Beschreibung	Standard	Beispiel
providerConfig.s3.skipCertValidation	SSL/TLS-Zertifikatsüberprüfung deaktivieren.	false	"true", "false"
providerConfig.s3.secure	Aktivieren Sie die sichere HTTPS-Kommunikation mit dem S3-Endpunkt.	true	"true", "false"
providerConfig.s3.proxyURL	Geben Sie die URL des Proxyservers an, der zur Verbindung mit S3 verwendet wird.	Keine	http://proxy.example.com:8080
providerConfig.s3.rootCA	Stellen Sie ein benutzerdefiniertes Root-CA-Zertifikat für die SSL/TLS-Verifizierung bereit.	Keine	"CN=MyCustomCA"
providerConfig.s3.useIAM	Aktivieren Sie die IAM-Authentifizierung für den Zugriff auf S3-Buckets. Gilt für EKS Pod Identity.	false	wahr, falsch

Informationen zu AppVault anzeigen

Sie können das Trident Protect CLI-Plugin verwenden, um Informationen über AppVault-Objekte anzuzeigen, 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) |
+-----+-----+-----+-----+
+-----+
```

- Optional können Sie den AppVaultPath für jede Ressource anzeigen, indem Sie das Flag `--show-paths` verwenden.

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

Entfernen Sie ein AppVault

Sie können ein AppVault-Objekt jederzeit entfernen.



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

Bevor Sie beginnen

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

Entfernen Sie ein AppVault mithilfe der Kubernetes-CLI

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

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

Entfernen Sie ein AppVault mit der Trident Protect CLI

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

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

Definieren Sie eine Anwendung für das Management 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 ein AppVault-CR erstellen, das bei der Durchführung von Datensicherungsoperationen an der Anwendung verwendet wird, und das AppVault-CR muss sich auf dem Cluster befinden, auf dem Trident Protect installiert ist. Das AppVault-CR ist spezifisch für Ihre Umgebung; Beispiele für AppVault-CRs finden Sie unter "[AppVault benutzerdefinierte Ressourcen](#)."

Eine Anwendung definieren

Sie müssen jede Anwendung, die Sie mit Trident Protect verwalten möchten, definieren. Sie können eine Anwendung zur Verwaltung entweder manuell durch Erstellen eines Anwendungs-CR oder mithilfe der Trident Protect CLI definieren.



Um eine VM-bezogene Anwendung zu definieren, die bestimmte KubeVirt virtuelle Maschinen (VMs) anhand ihres Namens anstatt ganzer Namensräume schützt, kann das `spec.includedVirtualMachines` Feld im Application-CR oder das `--virtual-machines` Flag in der CLI verwendet werden. Details, Feldreferenz und YAML-Beispiele sind unter "[Schutz von KubeVirt virtuellen Maschinen mit Trident Protect](#)" zu finden.

Fügen Sie eine Anwendung mithilfe eines CR hinzu

Schritte

1. Erstellen Sie die CR-Datei der Zielanwendung:

- a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) 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 Namen, den Sie wählen, da andere für Schutzvorgänge benötigte CR-Dateien auf diesen Wert verweisen.
 - **spec.includedNamespaces:** (*Erforderlich*) Verwenden Sie Namespace und Label-Selektor, um die Namespaces und Ressourcen anzugeben, die die Anwendung verwendet. Der Anwendungsnamespace 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-Scoped-Ressourcen anzugeben, die in die Anwendungsdefinition aufgenommen werden sollen. Mit diesem Attribut können Sie diese Ressourcen anhand ihrer Gruppe, Version, Art und Bezeichnungen auswählen.
 - **groupVersionKind:** (*Erforderlich*) Gibt die API-Gruppe, die Version und die 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 relevant, die von virtuellen Maschinen aus definiert werden, z. B. in KubeVirt-Umgebungen, in denen das Dateisystem vor Snapshots eingefroren wird. Legen Sie fest, ob diese Anwendung während eines Snapshots auf das Dateisystem schreiben darf. Ist die Option auf `true` gesetzt, ignoriert die Anwendung die globale Einstellung und kann während eines Snapshots auf das Dateisystem schreiben. Ist die Option auf `false` gesetzt, ignoriert die Anwendung die globale Einstellung und das Dateisystem wird während eines Snapshots eingefroren. Wird die Option angegeben, die Anwendung aber keine virtuellen Maschinen in der Anwendungsdefinition hat, wird die Annotation ignoriert. Wird sie nicht angegeben, folgt die Anwendung der ["globale Trident Protect freeze-Einstellung"](#).

Falls Sie diese Annotation nachträglich anwenden müssen, nachdem eine Anwendung bereits erstellt wurde, können Sie den folgenden Befehl verwenden:

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

+
Beispiel YAML:

+

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
```

```

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

```

1. (*Optional*) Falls erforderlich, können Sie der gleichen CR eine Ressourcenfilterung hinzufügen, um bestimmte Ressourcen einzuschließen oder auszuschließen:

- **Beispiel für einen generischen Filter:**

- **resourceFilter.resourceSelectionCriteria:** (Für die Filterung erforderlich) Verwenden Sie Include oder Exclude, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:
 - **resourceFilter.resourceMatchers:** Ein Array von resourceMatcher-Objekten. Wenn mehrere Elemente in diesem Array definiert werden, erfolgt die Übereinstimmung als ODER-Verknüpfung, und die Felder innerhalb jedes Elements (group, kind, version) werden als UND-Verknüpfung betrachtet.
 - **resourceMatchers[].group:** (*Optional*) Gruppe der zu filternden Ressource.
 - **resourceMatchers[].kind:** (*Optional*) Art der zu filternden Ressource.
 - **resourceMatchers[].version:** (*Optional*) Version der zu filternden Ressource.
 - **resourceMatchers[].names:** (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
 - **resourceMatchers[].namespaces:** (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
 - **resourceMatchers[].labelSelectors:** (*Optional*) Label-Selektorsequenz im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "[Kubernetes-](#)

Dokumentation". Beispiel: "trident.netapp.io/os=linux".



Wenn sowohl resourceFilter als auch labelSelector verwendet werden, wird resourceFilter zuerst ausgeführt und anschließend labelSelector 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"]
```

◦ **Beispiel für einen reinen PVC-Filter:**

Um eine reine PVC-Anwendung zu definieren, müssen Sie auch PersistentVolume und VolumeSnapshotClass im Ressourcenfilter angeben. Snapshot- und Backup-Vorgänge hängen von PersistentVolume (dem clusterweiten Volume, das an jedes PVC gebunden ist) und VolumeSnapshotClass (dem Snapshot-Treiber) ab und schlagen ohne diese fehl. Beispiel:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-pvc-app
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: my-app-namespace
  resourceFilter:
    resourceMatchers:
      - kind: PersistentVolumeClaim
        version: v1
      - kind: PersistentVolume
        version: v1
```

```
- kind: VolumeSnapshotClass
  version: v1
  resourceSelectionCriteria: Include
```

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, wobei Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Sie können Namensräume und Ressourcen in die Anwendungsdefinition einbinden, indem Sie durch Kommas getrennte Listen mit den in den Beispielen gezeigten Argumenten verwenden.

Beim Erstellen einer App können Sie optional eine Annotation verwenden, um festzulegen, ob die Anwendung während eines Snapshots auf das Dateisystem schreiben darf. Dies gilt nur für Anwendungen, die von virtuellen Maschinen definiert werden, beispielsweise in KubeVirt-Umgebungen, in denen das Dateisystem vor Snapshots eingefroren wird. Wenn Sie die Annotation auf `true` setzen, ignoriert die Anwendung die globale Einstellung und kann während eines Snapshots auf das Dateisystem schreiben. Wenn Sie sie auf `false` setzen, ignoriert die Anwendung 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 hat, wird die Annotation ignoriert. Wenn Sie die Annotation nicht verwenden, folgt die Anwendung der ["globale Trident Protect freeze-Einstellung"](#).

+ Um die Annotation bei der Erstellung einer Anwendung über die CLI anzugeben, können Sie das `--annotation` Flag verwenden.

+ * Die Anwendung wird erstellt und die globale Einstellung für das Einfrieren des Dateisystems verwendet:

+

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

+ * Die Anwendung wird erstellt und die lokale Anwendungseinstellung für das Verhalten beim Einfrieren des Dateisystems wird konfiguriert:

+

```
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 `--resource-filter-include` und `--resource-filter-exclude` Flags verwenden, um Ressourcen basierend auf `resourceSelectionCriteria` wie Gruppe, Art, Version, Bezeichnungen, Namen und Namensräumen einzuschließen oder auszuschließen, 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"]} ] '
```

Um eine reine PVC-Anwendung zu definieren, müssen auch `PersistentVolume` und `VolumeSnapshotClass` im Ressourcenfilter enthalten sein. Snapshot- und Backup-Vorgänge hängen von `PersistentVolume` (dem clusterweiten Volume, das an jedes PVC gebunden ist) und `VolumeSnapshotClass` (dem Snapshot-Treiber) ab und schlagen ohne diese fehl. Beispiel:

+

```
tridentctl-protect create app my-pvc-app --namespaces <my-app-
namespace> --resource-filter-include
' [{"Kind": "PersistentVolumeClaim", "Version": "v1"}, {"Kind": "PersistentVo
lume", "Version": "v1"}, {"Kind": "VolumeSnapshotClass", "Version": "v1"} ] '
-n <my-app-namespace>
```

Schützen Sie Anwendungen mit Trident Protect

Sie können alle von Trident Protect verwalteten Apps schützen, indem Sie Snapshots und Backups mithilfe einer automatisierten Datensicherungsstrategie oder nach Bedarf erstellen.



Sie können Trident Protect so konfigurieren, dass Dateisysteme während Datensicherungsoperationen 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-Namenspaces enthalten.

Erstellen Sie einen Snapshot mithilfe eines CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) 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 sinnvollen Namen für Ihre Umgebung.
 - **spec.applicationRef:** Der Kubernetes-Name der Anwendung, für die ein Snapshot erstellt werden soll.
 - **spec.appVaultRef:** (*Erforderlich*) Der Name des AppVault, wo die Snapshot-Inhalte (Metadaten) gespeichert werden sollen.
 - **spec.reclaimPolicy:** (*Optional*) Definiert, was mit dem AppArchive eines Snapshots geschieht, wenn die Snapshot-CR gelöscht wird. Das bedeutet, dass selbst wenn auf `Retain` gesetzt, der Snapshot gelöscht wird. 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 die `trident-protect-snapshot-cr.yaml` Datei mit den korrekten Werten gefüllt wurde, wird die CR angewendet:

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

Erstellen Sie einen Snapshot mithilfe der CLI

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 Backup-Vorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie in der ["AWS API-Dokumentation"](#).
- Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie in der ["AWS IAM-Dokumentation"](#).

Erstellen Sie eine Sicherung mithilfe eines CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) 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 sinnvollen Namen für Ihre Umgebung.
 - **spec.applicationRef:** (*Erforderlich*) Der Kubernetes-Name der zu sichernden Anwendung.
 - **spec.appVaultRef:** (*Erforderlich*) Der Name des AppVault, wo die Sicherungsinhalte gespeichert werden sollen.
 - **spec.dataMover:** (*Optional*) Eine Zeichenkette, die angibt, welches Sicherungstool für den Sicherungsvorgang verwendet werden soll. Mögliche Werte (Groß-/Kleinschreibung):
 - Restic
 - Kopia (Standard)
 - **spec.reclaimPolicy:** (*Optional*) Definiert, was mit einem Backup geschieht, wenn es aus seinem Anspruch entfernt wird. Mögliche Werte:
 - Delete
 - Retain (Standard)
 - **spec.snapshotRef:** (*Optional*): Name des Snapshots, der als Quelle für das Backup verwendet werden soll. Wenn kein Name angegeben wird, wird ein temporärer Snapshot erstellt und gesichert.

Beispiel YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
```

```

kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: Kopia

```

3. Nachdem die `trident-protect-backup-cr.yaml` Datei mit den korrekten Werten gefüllt wurde, wird die CR angewendet:

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

Erstellen Sie ein Backup mithilfe der CLI

Schritte

1. Erstellen Sie die Sicherungskopie, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. 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 das `--full-backup` Flag verwenden, um anzugeben, ob eine Sicherung nicht-inkrementell sein soll. Standardmäßig sind alle Backups inkrementell. Wenn dieses Flag verwendet wird, wird die Sicherung nicht-inkrementell. Es ist Best Practice, regelmäßig eine vollständige Sicherung durchzuführen und dazwischen inkrementelle Sicherungen zu erstellen, um das Risiko bei der Wiederherstellung zu minimieren.

Unterstützte Sicherungsanmerkungen

Die folgende Tabelle beschreibt die Anmerkungen, die Sie beim Erstellen eines Backup-CR verwenden können:

Anmerkung	Typ	Beschreibung	Standardwert
<code>protect.trident.netapp.io/full-backup</code>	Zeichenkette	Legt fest, ob eine Sicherung nicht inkrementell sein soll. Setzen Sie auf <code>true</code> , um eine nicht inkrementelle Sicherung zu erstellen. Es ist Best Practice, regelmäßig eine vollständige Sicherung durchzuführen und dazwischen inkrementelle Sicherungen zu erstellen, um das Risiko bei der Wiederherstellung zu minimieren.	"false"
<code>protect.trident.netapp.io/snapshots-hot-completion-timeout</code>	Zeichenkette	Die maximal zulässige Zeit für den Abschluss des gesamten Snapshot-Vorgangs.	"60m"

Anmerkung	Typ	Beschreibung	Standardwert
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	Zeichenkette	Die maximal zulässige Zeitspanne, bis Volume-Snapshots den einsatzbereiten Zustand erreichen.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	Zeichenkette	Die maximal zulässige Zeit für die Erstellung von Volume-Snapshots.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	Zeichenkette	Maximale Zeit (in Sekunden), die auf neu erstellte PersistentVolumeClaims (PVCs) gewartet wird, um die Bound Phase zu erreichen, bevor der Vorgang fehlschlägt.	"1200" (20 Minuten)

Erstellen Sie einen Zeitplan für die Datensicherung

Eine Datensicherungsstrategie schützt eine App, indem sie Snapshots, Backups oder beides nach einem definierten Zeitplan erstellt. Sie können wählen, Snapshots und Backups stündlich, täglich, wöchentlich und monatlich zu erstellen, und Sie können die Anzahl der aufzubewahrenden Kopien angeben. Sie können ein nicht-inkrementelles vollständiges Backup mit der Annotation `full-backup-rule` planen. Standardmäßig sind alle Backups inkrementell. Das periodische Durchführen eines vollständigen Backups zusammen mit inkrementellen Backups dazwischen hilft, das Risiko bei Wiederherstellungen zu reduzieren.



- Sie können Zeitpläne für Snapshots nur erstellen, indem Sie `backupRetention` auf Null und `snapshotRetention` auf einen Wert größer als Null setzen. Das Setzen von `snapshotRetention` auf Null bedeutet, dass bei geplanten Backups zwar weiterhin Snapshots erstellt werden, diese jedoch 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) 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 sinnvollen Namen für Ihre Umgebung.
 - **spec.dataMover:** (*Optional*) Eine Zeichenkette, die angibt, welches Sicherungstool für den Sicherungsvorgang verwendet werden soll. Mögliche Werte (Groß-/Kleinschreibung):
 - `Restic`
 - `Kopia` (Standard)
 - **spec.applicationRef:** Der Kubernetes-Name der zu sichernden Anwendung.
 - **spec.appVaultRef:** (*Erforderlich*) Der Name des AppVault, wo die Sicherungsinhalte gespeichert werden sollen.
 - **spec.backupRetention:** (*Erforderlich*) Die Anzahl der aufzubewahrenden Backups. Null bedeutet, dass keine Backups erstellt werden sollen (nur Snapshots).

- **spec.backupReclaimPolicy:** (*Optional*) Legt fest, was mit einem Backup geschieht, wenn die Backup-CR während der Aufbewahrungsfrist gelöscht wird. Nach Ablauf der Aufbewahrungsfrist werden Backups immer gelöscht. Mögliche Werte (Groß-/Kleinschreibung):
 - Retain (Standard)
 - Delete
- **spec.snapshotRetention:** (*Erforderlich*) Die Anzahl der aufzubewahrenden Snapshots. Zero bedeutet, dass keine Snapshots erstellt werden sollen.
- **spec.snapshotReclaimPolicy:** (*Optional*) Legt fest, was mit einem Snapshot geschieht, wenn die Snapshot-CR während seiner Aufbewahrungsfrist gelöscht wird. Nach Ablauf der Aufbewahrungsfrist werden Snapshots immer gelöscht. Mögliche Werte (Groß-/Kleinschreibung):
 - Retain
 - Delete (Standard)
- **spec.granularity:** Die Häufigkeit, mit der der Zeitplan ausgeführt werden soll. Mögliche Werte sowie zugehörige Pflichtfelder:
 - Hourly (erfordert, dass Sie `spec.minute` angeben)
 - Daily (erfordert, dass Sie `spec.minute` und `spec.hour` angeben)
 - Weekly (erfordert, dass Sie `spec.minute`, `spec.hour` angeben, und `spec.dayOfWeek`)
 - Monthly (erfordert, dass Sie `spec.minute`, `spec.hour` angeben, und `spec.dayOfMonth`)
 - Custom
- **spec.dayOfMonth:** (*Optional*) Der Tag des Monats (1 - 31), an dem der Zeitplan ausgeführt werden soll. Dieses Feld ist erforderlich, wenn die Granularität auf `Monthly` eingestellt ist. Der Wert muss als Zeichenkette angegeben werden.
- **spec.dayOfWeek:** (*Optional*) Der Wochentag (0 - 7), an dem der Zeitplan ausgeführt werden soll. Werte von 0 oder 7 bedeuten Sonntag. Dieses Feld ist erforderlich, wenn die Granularität auf `Weekly` eingestellt ist. Der Wert muss als Zeichenkette 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 `Daily`, `Weekly` oder `Monthly` eingestellt ist. Der Wert muss als Zeichenkette 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 `Hourly`, `Daily`, `Weekly` oder `Monthly` gesetzt ist. Der Wert muss als Zeichenkette angegeben werden.
- **spec.runImmediately:** (*Optional*) Auf `true` setzen, um beim Erstellen des Zeitplans einen einmaligen, sofortigen Baseline-Lauf (Backup und/oder Snapshot gemäß den Aufbewahrungseinstellungen) auszulösen. Standardmäßig `false`. Dies ändert die nachfolgende Wiederholung nicht.

Beispiel-YAML für Backup- und Snapshot-Zeitplan:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
```

```

namespace: my-app-namespace
name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: Daily
  hour: "0"
  minute: "0"

```

Beispiel-YAML für Snapshot-only-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"

```

Beispiel-YAML für einen Zeitplan mit sofortiger Ausführung:

```

---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-daily-schedule-run-immediately
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "7"
  snapshotRetention: "7"
  granularity: Daily
  hour: "3"

```

```
minute: "0"  
runImmediately: true
```

3. Nachdem die `trident-protect-schedule-cr.yaml` Datei mit den korrekten Werten gefüllt wurde, wird die CR angewendet:

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

Erstellen Sie einen Zeitplan mithilfe der CLI

Schritte

1. Erstellen Sie den Schutzzeitplan, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Beispiel:



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

Die folgenden Optionen bieten zusätzliche Kontrolle über Ihren Zeitplan:

- **Planung vollständiger Backups:** Verwenden Sie das `--full-backup-rule`-Flag, um nicht-inkrementelle vollständige Backups zu planen. Dieses Flag funktioniert nur mit `--granularity Daily`. Mögliche Werte:
 - **Always:** Erstellen Sie jeden Tag ein vollständiges Backup.
 - **Bestimmte Wochentage:** Geben Sie einen oder mehrere Tage durch Kommas getrennt an (zum Beispiel "Monday, Thursday"). Gültige Werte: Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag, Sonntag.



Die `--full-backup-rule` Kennzeichnung funktioniert nicht bei stündlicher, wöchentlicher oder monatlicher Granularität.

- **Sofortiger Basisdatenschutz:** Verwenden Sie `--run-immediately true`, um sofort bei der Erstellung des Zeitplans ein erstes Backup oder einen Snapshot zu erstellen, anstatt auf die erste geplante Ausführung zu warten. Standardmäßig ist `false`.
- **Snapshot-only-Zeitpläne:** Setzen Sie `--backup-retention 0` und geben Sie einen Wert größer als null für `--snapshot-retention` an.

Unterstützte Zeitplananmerkungen

Die folgende Tabelle beschreibt die Anmerkungen, die Sie beim Erstellen eines Schedule-CR verwenden können:

Anmerkung	Typ	Beschreibung	Standardwert
<code>protect.trident.netapp.io/full-backup-rule</code>	Zeichenkette	Legt die Regel für die Planung vollständiger Backups fest. Sie können es auf <code>Always</code> für konstante vollständige Sicherungen festlegen oder es basierend auf Ihren Anforderungen anpassen. Wenn Sie beispielsweise die tägliche Granularität wählen, können Sie die Wochentage angeben, an denen die vollständige Sicherung erfolgen soll (zum Beispiel <code>"Monday, Thursday"</code>). Gültige Wochentage sind: Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag, Sonntag. Beachten Sie, dass diese Annotation nur mit Zeitplänen verwendet werden kann, die <code>granularity</code> auf <code>Daily</code> gesetzt sind.	Nicht festgelegt (alle Backups sind inkrementell)
<code>protect.trident.netapp.io/snapshot-completion-timeout</code>	Zeichenkette	Die maximal zulässige Zeit für den Abschluss des gesamten Snapshot-Vorgangs.	"60m"
<code>protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout</code>	Zeichenkette	Die maximal zulässige Zeitspanne, bis Volume-Snapshots den einsatzbereiten Zustand erreichen.	"30m"
<code>protect.trident.netapp.io/volume-snapshots-created-timeout</code>	Zeichenkette	Die maximal zulässige Zeit für die Erstellung von Volume-Snapshots.	"5m"
<code>protect.trident.netapp.io/pvc-bind-timeout-sec</code>	Zeichenkette	Maximale Zeit (in Sekunden), die auf neu erstellte PersistentVolumeClaims (PVCs) gewartet wird, um die <code>Bound Phase</code> zu erreichen, bevor der Vorgang fehlschlägt.	"1200" (20 Minuten)

Einen Snapshot löschen

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

Schritte

1. Entfernen Sie die mit dem Snapshot verknüpfte Snapshot-CR:

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

Ein Backup löschen

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



Stellen Sie sicher, dass die Rückgewinnungsrichtlinie auf `Delete` gesetzt ist, um alle Sicherungsdaten aus dem Objektspeicher zu entfernen. Die Standardeinstellung der Richtlinie ist `Retain`, um versehentlichen Datenverlust zu vermeiden. Wenn die Richtlinie nicht auf `Delete` geändert wird, verbleiben die Sicherungsdaten im Objektspeicher und müssen manuell gelöscht werden.

Schritte

1. Entfernen Sie die mit dem Backup verknüpfte Backup-CR:

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

Überprüfen Sie den Status eines Sicherungsvorgangs

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

Schritte

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

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

Backup und Restore für azure-netapp-files (ANF)-Vorgänge aktivieren

Wenn Sie Trident Protect installiert haben, können Sie die platzsparende Sicherungs- und Wiederherstellungsfunktion 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 verfügt nur über eingeschränkte Schutzfunktionen, bis Sie dieses Verfahren abgeschlossen haben.
- Sie haben `azure-netapp-files` als Standard-Speicherklasse für Ihr Speicher-Backend ausgewählt.

▼ Für Konfigurationsschritte erweitern

1. Führen Sie Folgendes in Trident 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, wählt Trident Protect die reguläre Sicherungsfunktion, die während des Sicherungsvorgangs temporär 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 Backup und Restore mit Trident Protect bereit. Jede PVC kann auch von anderen Anwendungen für Backups und Restores 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 Ausführungs-Hook nach der Wiederherstellung vorhanden ist, wird er automatisch als Teil des Wiederherstellungsvorgangs ausgeführt.
- Die Wiederherstellung aus einem Backup in einen anderen Namespace oder in den ursprünglichen Namespace wird für qtree Volumes unterstützt. Die Wiederherstellung aus einem Snapshot in einen anderen Namespace oder in den ursprünglichen Namespace wird für qtree Volumes jedoch nicht unterstützt.
- Sie können die Wiederherstellungsvorgänge mithilfe erweiterter Einstellungen anpassen. Weitere Informationen sind unter ["Verwenden Sie die erweiterten Trident Protect-Wiederherstellungseinstellungen"](#) zu finden.

- Um bestimmte KubeVirt virtuelle Maschinen (VMs) aus einem Archiv wiederherzustellen, ohne alle Namespace-Ressourcen wiederherzustellen, kann das `includedVirtualMachines` Feld in einem Restore-CR verwendet werden. Unterstützte CRs, Feldreferenzen und YAML-Beispiele sind unter ["Schutz von KubeVirt virtuellen Maschinen mit Trident Protect"](#) zu finden.

Wiederherstellung aus einem Backup in einen anderen Namensraum

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



- Die Wiederherstellung einer Sicherung in einem anderen Namensraum mit vorhandenen Ressourcen ändert keine Ressourcen, die denselben Namen wie die in der Sicherung haben. Um alle Ressourcen in der Sicherung wiederherzustellen, löschen und erstellen Sie entweder den Zielnamensraum neu oder stellen Sie die Sicherung in einem neuen Namensraum wieder her.
- Wenn Sie eine CR zur Wiederherstellung in einem neuen Namespace verwenden, müssen Sie den Ziel-Namespace manuell erstellen, bevor Sie die CR anwenden. Trident Protect erstellt Namespaces automatisch nur bei Verwendung der CLI.

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger dauernden s3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie in der ["AWS API-Dokumentation"](#).
- Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie in der ["AWS IAM-Dokumentation"](#).



Wenn Sie Backups mit Kopia als Data Mover wiederherstellen, können Sie optional Anmerkungen in der CR oder über die CLI angeben, um das Verhalten des von Kopia verwendeten temporären Speichers zu steuern. Weitere Informationen über die Optionen, die Sie konfigurieren können, finden Sie in der ["Kopia-Dokumentation"](#). Verwenden Sie den `tridentctl-protect create --help`-Befehl, um weitere Informationen zum Angeben von Anmerkungen mit der Trident Protect CLI zu erhalten.

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) 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 sinnvollen Namen für Ihre Umgebung.
 - **spec.appArchivePath:** Der Pfad innerhalb von AppVault, in dem die Sicherungsinhalte gespeichert sind. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

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

- **spec.appVaultRef:** (*Erforderlich*) Der Name des AppVault, in dem die Sicherungsinhalte gespeichert sind.
- **spec.destinationApplicationName:** (*Optional*) Der Name für die wiederhergestellte Anwendung. Falls angegeben, verwendet die wiederhergestellte Anwendung diesen Namen. Falls nicht angegeben, verwendet die wiederhergestellte Anwendung den Namen der Quellanwendung.
- **spec.namespaceMapping:** Die Zuordnung des Quell-Namespaces des Wiederherstellungsvorgangs zum Ziel-Namespace. Ersetzen Sie my-source-namespace und my-destination-namespace durch 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
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace",
"destination": "my-destination-namespace"}]
```

3. (*Optional*) Falls nur bestimmte Ressourcen der Anwendung für die Wiederherstellung ausgewählt werden sollen, kann ein Filter hinzugefügt werden, der Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließt:



Trident Protect wählt bestimmte Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn beispielsweise eine Ressource für einen Persistent Volume Claim ausgewählt wird 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, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:
 - **resourceFilter.resourceMatchers:** Ein Array von resourceMatcher-Objekten. Wenn mehrere Elemente in diesem Array definiert werden, erfolgt die Übereinstimmung als ODER-Verknüpfung, und die Felder innerhalb jedes Elements (group, kind, version) werden als UND-Verknüpfung betrachtet.
 - **resourceMatchers[].group:** (*Optional*) Gruppe der zu filternden Ressource.
 - **resourceMatchers[].kind:** (*Optional*) Art der zu filternden Ressource.
 - **resourceMatchers[].version:** (*Optional*) Version der zu filternden Ressource.

- **resourceMatchers[].names:** (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].namespaces:** (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].labelSelectors:** (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "[Kubernetes-Dokumentation](#)". 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 die trident-protect-backup-restore-cr.yaml Datei mit den korrekten Werten gefüllt wurde, wird die CR angewendet:

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

Verwenden Sie die Befehlszeile

Schritte

1. Stellen Sie die Sicherung in einem anderen Namensraum wieder her, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Das namespace-mapping Argument verwendet durch Doppelpunkte getrennte Namensräume, um Quell-Namensräume den korrekten Ziel-Namensräumen im Format source1:dest1, source2:dest2 zuzuordnen. Beispiel:

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

Stellen Sie aus einer Sicherung in den ursprünglichen Namensraum wieder her

Sie können eine Sicherung jederzeit im ursprünglichen Namensraum wiederherstellen. Wenn Sie eine Wiederherstellung vor Ort durchführen, verwaltet Trident Protect automatisch Schutzzeitpläne und laufende Vorgänge, um ungültige Wiederherstellungspunkte zu verhindern:

- Alle für die Anwendung aktivierten Schutzzeitpläne werden vor Beginn der Wiederherstellung deaktiviert. Dadurch wird verhindert, dass geplante Sicherungen oder Snapshots ausgeführt werden, während die Anwendungsressourcen wiederhergestellt werden.
- Nach erfolgreicher Wiederherstellung werden nur die vor der Wiederherstellung aktivierten Zeitpläne wieder aktiviert. Bereits deaktivierte Zeitpläne bleiben deaktiviert.
- Laufende Sicherungs- oder Snapshot-Vorgänge werden vor Beginn der Wiederherstellung abgebrochen. Wird ein Vorgang nicht innerhalb von 5 Minuten abgebrochen, wird die Wiederherstellung fortgesetzt und eine Warnung im Wiederherstellungs-CR-Status protokolliert.

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger dauernden s3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie in der ["AWS API-Dokumentation"](#).
- Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie in der ["AWS IAM-Dokumentation"](#).



Wenn Sie Backups mit Kopia als Data Mover wiederherstellen, können Sie optional Anmerkungen in der CR oder über die CLI angeben, um das Verhalten des von Kopia verwendeten temporären Speichers zu steuern. Weitere Informationen über die Optionen, die Sie konfigurieren können, finden Sie in der ["Kopia-Dokumentation"](#). Verwenden Sie den `tridentctl-protect create --help`-Befehl, um weitere Informationen zum Angeben von Anmerkungen mit der Trident Protect CLI zu erhalten.

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) 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 sinnvollen Namen für Ihre Umgebung.
 - **spec.appArchivePath:** Der Pfad innerhalb von AppVault, in dem die Sicherungsinhalte gespeichert sind. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

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

- **spec.appVaultRef:** (*Erforderlich*) Der Name des AppVault, 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 nur bestimmte Ressourcen der Anwendung für die Wiederherstellung ausgewählt werden sollen, kann ein Filter hinzugefügt werden, der Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließt:



Trident Protect wählt bestimmte Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn beispielsweise eine Ressource für einen Persistent Volume Claim ausgewählt wird 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, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:
 - **resourceFilter.resourceMatchers:** Ein Array von resourceMatcher-Objekten. Wenn mehrere Elemente in diesem Array definiert werden, erfolgt die Übereinstimmung als ODER-Verknüpfung, und die Felder innerhalb jedes Elements (group, kind, version) werden als UND-Verknüpfung betrachtet.
 - **resourceMatchers[].group:** (*Optional*) Gruppe der zu filternden Ressource.
 - **resourceMatchers[].kind:** (*Optional*) Art der zu filternden Ressource.
 - **resourceMatchers[].version:** (*Optional*) Version der zu filternden Ressource.
 - **resourceMatchers[].names:** (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
 - **resourceMatchers[].namespaces:** (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
 - **resourceMatchers[].labelSelectors:** (*Optional*) Label-Selektorenzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "[Kubernetes-Dokumentation](#)". 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 die `trident-protect-backup-ipr-cr.yaml` Datei mit den korrekten Werten gefüllt wurde, wird die CR angewendet:

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

Verwenden Sie die Befehlszeile

Schritte

1. Stellen Sie die Sicherung im ursprünglichen Namensraum wieder her, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Das `backup` Argument verwendet einen Namensraum und einen Sicherungsnamen im 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 ein Backup auf einem anderen Cluster wiederherstellen, wenn es ein Problem mit dem ursprünglichen Cluster gibt.



- Wenn Sie Backups mit Kopia als Data Mover wiederherstellen, können Sie optional Anmerkungen in der CR oder über die CLI angeben, um das Verhalten des von Kopia verwendeten temporären Speichers zu steuern. Weitere Informationen über die Optionen, die Sie konfigurieren können, finden Sie in der "[Kopia-Dokumentation](#)". Verwenden Sie den `tridentctl-protect create --help`-Befehl, um weitere Informationen zum Angeben von Anmerkungen mit der Trident Protect CLI zu erhalten.
- Wenn Sie eine CR zur Wiederherstellung in einem neuen Namespace verwenden, müssen Sie den Ziel-Namespace manuell erstellen, bevor Sie die CR anwenden. Trident Protect erstellt Namespaces automatisch nur bei Verwendung der CLI.

Bevor Sie beginnen

Stellen Sie sicher, dass die folgenden Voraussetzungen erfüllt sind:

- Auf dem Ziel-Cluster ist Trident Protect installiert.
- Der Ziel-Cluster hat Zugriff auf den Bucket-Pfad desselben AppVault wie der Quell-Cluster, in dem die Sicherung gespeichert ist.
- Stellen Sie sicher, dass Ihre lokale Umgebung eine Verbindung zum im AppVault CR definierten Objektspeicher-Bucket herstellen kann, wenn Sie den `tridentctl-protect get appvaultcontent` Befehl ausführen. Wenn Netzwerkbeschränkungen den Zugriff verhindern, führen Sie die Trident Protect CLI stattdessen innerhalb eines Pods auf dem Ziel-Cluster aus.
- Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger dauernden Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.
 - Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie in der ["AWS API-Dokumentation"](#).
 - Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie in der ["AWS-Dokumentation"](#).

Schritte

1. Überprüfen Sie, ob die AppVault CR auf dem Ziel-Cluster mithilfe des Trident Protect CLI-Plugins vorhanden ist:

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



Falls die AppVault CR auf dem Ziel-Cluster nicht existiert, erstellen Sie sie gemäß den Schritten in ["Verwenden Sie Trident Protect AppVault-Objekte, um Buckets zu verwalten"](#).

2. Sehen Sie sich die Sicherungsinhalte des verfügbaren AppVault auf dem Ziel-Cluster an und notieren Sie sich `appArchivePath` der Sicherung, die Sie wiederherstellen möchten:

```
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, entsprechenden Anwendungsnamen, Zeitstempel und Archivpfade.

Beispielausgabe:

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

```
| production1 | wordpress | backup | wordpress-bkup-2 | 2024-10-30
08:37:40 (UTC) | backuppath2 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

3. Stellen Sie die Anwendung im Ziel-Cluster mithilfe des AppVault-Namens und des Archivpfads wieder her:



Bei Verwendung eines CR muss sichergestellt werden, dass der für die Anwendungswiederherstellung vorgesehene Namespace auf dem Ziel-Cluster existiert.

Verwenden Sie einen CR

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) 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 sinnvollen Namen für Ihre Umgebung.
 - **spec.appVaultRef:** (*Erforderlich*) Der Name des AppVault, in dem die Sicherungsinhalte gespeichert sind.
 - **spec.appArchivePath:** (*Erforderlich*) Der Pfad innerhalb von AppVault, in dem die Sicherungsinhalte gespeichert sind. Verwenden Sie den Befehl aus Schritt 2, um die Sicherungsinhalte anzuzeigen und `appArchivePath` für die Sicherung zu finden, die Sie wiederherstellen möchten.
 - **spec.destinationApplicationName:** (*Optional*) Der Name für die wiederhergestellte Anwendung. Falls angegeben, verwendet die wiederhergestellte Anwendung diesen Namen. Falls nicht angegeben, verwendet die wiederhergestellte Anwendung den Namen der Quellanwendung.
 - **spec.namespaceMapping:** Die Zuordnung des Quell-Namespace des Wiederherstellungsvorgangs zum Ziel-Namespace. Ersetzen Sie `my-source-namespace` und `my-destination-namespace` durch 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
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. Nachdem die `trident-protect-backup-restore-cr.yaml` Datei mit den korrekten Werten

gefüllt wurde, wird die CR angewendet:

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

Verwenden Sie die Befehlszeile

1. Verwenden Sie den folgenden Befehl, um die Anwendung wiederherzustellen, und ersetzen Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung. Das Argument namespace-mapping verwendet durch Doppelpunkte getrennte Namensräume, um Quell-Namensräume den korrekten Ziel-Namensräumen im Format Quelle1:Ziel1,Quelle2:Ziel2 zuzuordnen. Beispiel:

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

Wiederherstellung aus einem Snapshot in einen anderen Namespace

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



- SnapshotRestore unterstützt das `spec.storageClassMapping` Attribut, jedoch nur, wenn die Quell- und Ziel-Speicherklassen dasselbe Speicher-Backend verwenden. Wenn Sie versuchen, auf eine `StorageClass` wiederherzustellen, die ein anderes Speicher-Backend verwendet, schlägt der Wiederherstellungsvorgang fehl.
- Wenn Sie eine CR zur Wiederherstellung in einem neuen Namespace verwenden, müssen Sie den Ziel-Namespace manuell erstellen, bevor Sie die CR anwenden. Trident Protect erstellt Namespaces automatisch nur bei Verwendung der CLI.

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger dauernden s3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie in der ["AWS API-Dokumentation"](#).
- Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie in der ["AWS IAM-Dokumentation"](#).

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) 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 sinnvollen Namen für Ihre Umgebung.
 - **spec.appVaultRef:** (*Erforderlich*) Der Name des AppVault, in dem die Snapshot-Inhalte gespeichert sind.
 - **spec.appArchivePath:** Der Pfad innerhalb von AppVault, in dem die Snapshot-Inhalte gespeichert sind. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

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

- **spec.destinationApplicationName:** (*Optional*) Der Name für die wiederhergestellte Anwendung. Falls angegeben, verwendet die wiederhergestellte Anwendung diesen Namen. Falls nicht angegeben, verwendet die wiederhergestellte Anwendung den Namen der Quellanwendung.
- **spec.namespaceMapping:** Die Zuordnung des Quell-Namespace des Wiederherstellungsvorgangs zum Ziel-Namespace. Ersetzen Sie `my-source-namespace` und `my-destination-namespace` durch Informationen aus Ihrer Umgebung.

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

3. (*Optional*) Falls nur bestimmte Ressourcen der Anwendung für die Wiederherstellung ausgewählt werden sollen, kann ein Filter hinzugefügt werden, der Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließt:



Trident Protect wählt bestimmte Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn beispielsweise eine Ressource für einen Persistent Volume Claim ausgewählt wird 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, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:

- **resourceFilter.resourceMatchers:** Ein Array von resourceMatcher-Objekten. Wenn mehrere Elemente in diesem Array definiert werden, erfolgt die Übereinstimmung als ODER-Verknüpfung, und die Felder innerhalb jedes Elements (group, kind, version) werden als UND-Verknüpfung betrachtet.
 - **resourceMatchers[].group:** (*Optional*) Gruppe der zu filternden Ressource.
 - **resourceMatchers[].kind:** (*Optional*) Art der zu filternden Ressource.
 - **resourceMatchers[].version:** (*Optional*) Version der zu filternden Ressource.
 - **resourceMatchers[].names:** (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
 - **resourceMatchers[].namespaces:** (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
 - **resourceMatchers[].labelSelectors:** (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "[Kubernetes-Dokumentation](#)". 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 die trident-protect-snapshot-restore-cr.yaml Datei mit den korrekten Werten gefüllt wurde, wird die CR angewendet:

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

Verwenden Sie die Befehlszeile
Schritte

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

- Das `snapshot` Argument verwendet einen Namespace und einen Snapshot-Namen im Format `<namespace>/<name>`.
- Das `namespace-mapping` Argument verwendet durch Doppelpunkte getrennte Namensräume, um Quell-Namensräume den korrekten Ziel-Namensräumen im Format `source1:dest1,source2:dest2` zuzuordnen.

Beispiel:

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

Wiederherstellung aus einem Snapshot in den ursprünglichen Namensraum

Sie können einen Snapshot jederzeit im ursprünglichen Namensraum wiederherstellen. Wenn Sie eine Wiederherstellung vor Ort durchführen, verwaltet Trident Protect automatisch Schutzzeitpläne und laufende Vorgänge, um ungültige Wiederherstellungspunkte zu verhindern:

- Alle für die Anwendung aktivierten Schutzzeitpläne werden vor Beginn der Wiederherstellung deaktiviert. Dadurch wird verhindert, dass geplante Sicherungen oder Snapshots ausgeführt werden, während die Anwendungsressourcen wiederhergestellt werden.
- Nach erfolgreicher Wiederherstellung werden nur die vor der Wiederherstellung aktivierten Zeitpläne wieder aktiviert. Bereits deaktivierte Zeitpläne bleiben deaktiviert.
- Laufende Sicherungs- oder Snapshot-Vorgänge werden vor Beginn der Wiederherstellung abgebrochen. Wird ein Vorgang nicht innerhalb von 5 Minuten abgebrochen, wird die Wiederherstellung fortgesetzt und eine Warnung im Wiederherstellungs-CR-Status protokolliert.

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger dauernden s3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie in der ["AWS API-Dokumentation"](#).
- Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie in der ["AWS IAM-Dokumentation"](#).

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) 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 sinnvollen Namen für Ihre Umgebung.
- **spec.appVaultRef:** (*Erforderlich*) Der Name des AppVault, in dem die Snapshot-Inhalte gespeichert sind.
- **spec.appArchivePath:** Der Pfad innerhalb von AppVault, in dem die Snapshot-Inhalte gespeichert sind. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

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

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

+ . (*Optional*) Falls nur bestimmte Ressourcen der Anwendung für die Wiederherstellung ausgewählt werden sollen, kann ein Filter hinzugefügt werden, der Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließt:

+ HINWEIS: Trident Protect wählt einige Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn beispielsweise eine Ressource für einen Persistent Volume Claim ausgewählt wird und diese einen zugehörigen Pod hat, stellt Trident Protect auch den zugehörigen Pod wieder her.

+ * **resourceFilter.resourceSelectionCriteria:** (Für die Filterung erforderlich) Include oder Exclude verwenden, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Die folgenden resourceMatchers-Parameter hinzufügen, um die einzuschließenden oder auszuschließenden Ressourcen zu definieren: **resourceFilter.resourceMatchers:** Ein Array von resourceMatcher-Objekten. Wenn mehrere Elemente in diesem Array definiert werden, erfolgt die Übereinstimmung als ODER-Verknüpfung, und die Felder innerhalb jedes Elements (group, kind, version) werden als UND-Verknüpfung betrachtet. * resourceMatchers[].group: (*Optional*) Gruppe der zu filternden Ressource. * resourceMatchers[].kind: (*Optional*) Art der zu filternden Ressource. * resourceMatchers[].version: (*Optional*) Version der zu filternden Ressource. * resourceMatchers[].names: (*Optional*) Namen im Kubernetes-Feld metadata.name der zu filternden Ressource. * resourceMatchers[].namespaces: (*Optional*) Namespaces im Kubernetes-Feld metadata.name der zu filternden Ressource. * resourceMatchers[].labelSelectors: (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Feld metadata.name der Ressource, wie in der "[Kubernetes-Dokumentation](#)" definiert. Beispiel: "trident.netapp.io/os=linux".

+ Zum 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"]
```

+ . Nachdem die trident-protect-snapshot-ipr-cr.yaml Datei mit den korrekten Werten gefüllt wurde, wird die CR angewendet:

+

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

Verwenden Sie die Befehlszeile

Schritte

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

```
tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <namespace/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, und ersetzen Sie 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 Annotationen, Namespace-Einstellungen und Speicheroptionen an Ihre spezifischen Anforderungen anpassen.

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

Während Wiederherstellungs- und Failover-Vorgängen werden die Labels und Annotationen im Ziel-Namensraum so angepasst, dass sie den Labels und Annotationen im Quell-Namensraum entsprechen. Labels oder Annotationen aus dem Quell-Namensraum, die im Ziel-Namensraum nicht existieren, werden hinzugefügt, und alle Labels oder Annotationen, die bereits vorhanden sind, 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-Annotationen stellen sicher, dass wiederhergestellte Pods die entsprechenden Berechtigungen und Sicherheitskonfigurationen einhalten, die durch OpenShift Security Context Constraints (SCCs) definiert sind, und ohne Berechtigungsprobleme auf Volumes zugreifen können. Weitere Informationen finden sich in der ["OpenShift security context constraints Dokumentation"](#).

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

```
helm upgrade trident-protect -n trident-protect netapp-trident-protect/trident-protect \
  --set-string
  restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_key_to_skip_2>}" \
  --reuse-values
```



Bei der Durchführung einer Wiederherstellungs- oder Failover-Operation werden alle in `restoreSkipNamespaceAnnotations` und `restoreSkipNamespaceLabels` angegebenen Namespace-Annotationen und -Labels von der Wiederherstellungs- oder Failover-Operation ausgeschlossen. Stellen Sie sicher, dass diese Einstellungen während der initialen Helm-Installation konfiguriert sind. Weitere Informationen sind unter ["Konfigurieren Sie zusätzliche Trident Protect Helm-Chart-Einstellungen"](#) zu finden.

Wenn Sie die Quellanwendung mit Helm mit dem `--create-namespace` Flag installiert haben, wird dem `name` Label-Schlüssel eine besondere Behandlung zuteil. Während des Wiederherstellungs- oder Failover-Prozesses kopiert Trident Protect dieses Label in den Ziel-Namespace, aktualisiert jedoch den Wert auf den Wert des Ziel-Namespace, wenn der Wert aus der Quelle mit dem Quell-Namespace übereinstimmt. Wenn dieser Wert nicht mit dem Quell-Namespace übereinstimmt, wird er unverändert in den Ziel-Namespace kopiert.

Beispiel

Das folgende Beispiel zeigt einen Quell- und einen Ziel-Namensraum mit jeweils unterschiedlichen Annotationen und Labels. Sie können den Zustand des Ziel-Namensraums vor und nach der Operation sehen 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	Etiketten
Namespace ns-1 (Quelle)	<ul style="list-style-type: none">• annotation.one/key: "updatedvalue"• annotation.two/key: "true"	<ul style="list-style-type: none">• environment=production• compliance=hipaa• name=ns-1
Namespace ns-2 (Ziel)	<ul style="list-style-type: none">• annotation.one/key: "true"• annotation.three/key: "false"	<ul style="list-style-type: none">• role=database

Nach dem Wiederherstellungsvorgang

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

Namensraum	Anmerkungen	Etiketten
Namespace ns-2 (Ziel)	<ul style="list-style-type: none">• annotation.one/key: "updatedvalue"• annotation.two/key: "true"• annotation.three/key: "false"	<ul style="list-style-type: none">• name=ns-2• compliance=hipaa• environment=production• role=database

Unterstützte Felder

In diesem Abschnitt werden zusätzliche Felder beschrieben, die für Wiederherstellungsvorgänge zur Verfügung stehen.

Speicherklassenzuordnung

Das `spec.storageClassMapping` Attribut definiert eine Zuordnung von einer Speicherklasse in der Quellanwendung zu einer neuen Speicherklasse im Zielcluster. Sie können dies verwenden, wenn Sie Anwendungen zwischen Clustern mit unterschiedlichen Speicherklassen migrieren oder das Speicher-Backend für BackupRestore-Operationen 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	Zeichenkette	Die maximal zulässige Zeit (in Sekunden), in der der Datenübertragungsvorgang angehalten werden darf.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	Zeichenkette	Die maximale Größenbeschränkung (in Megabytes) für den Kopia-Inhaltscache.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	Zeichenkette	Maximale Zeit (in Sekunden), die auf neu erstellte PersistentVolumeClaims (PVCs) gewartet wird, um die Bound Phase zu erreichen, bevor der Vorgang fehlschlägt. Gilt für alle Restore-CR-Typen (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Ein höherer Wert ist sinnvoll, wenn das Storage-Backend oder der Cluster häufig mehr Zeit benötigt.	"1200" (20 Minuten)

Ändern Sie Ressourcen während der Wiederherstellung mithilfe benutzerdefinierter Ressourcen

Ressourcentransformationen ermöglichen die Modifikation einer Ressource während der Wiederherstellung. Dies ist hilfreich, wenn sich die wiederhergestellte Version von der ursprünglichen unterscheiden muss, zum Beispiel beim Aktualisieren eines Hostnamens in einer ConfigMap, wenn eine Anwendung in einem anderen Cluster wiederhergestellt wird.

Wie Ressourcenmodifikation funktioniert

Das `transformations` Feld in `SnapshotRestore`, `BackupRestore`, `AppMirrorRelationship` und anderen Wiederherstellungsressourcen ermöglicht die Anpassung von Kubernetes-Ressourcen während des Wiederherstellungsprozesses. Dies ist hilfreich, wenn die wiederhergestellte Anwendung anders als das Original konfiguriert werden soll, beispielsweise durch Änderungen an Hostnamen, Registry-URLs, Ressourcenlimits oder Umgebungsvariablen.

Ressourcentransformationen verwenden ["RFC 6902 JSON Patch"](#) Operationen und ["RFC 6901 JSON Pointer"](#) Pfade, um bestimmte Felder innerhalb von Kubernetes-Ressourcen gezielt anzusprechen und zu modifizieren.

Nachfolgend ist die grundlegende Struktur eines Wiederherstellungsobjekts dargestellt, das Ressourcentransformationen enthält:

```

apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: my-restore
  namespace: target-namespace
spec:
  appVaultRef: my-vault
  appArchivePath: /path/to/snapshot
  namespaceMapping:
    - source: source-ns
      destination: target-ns
  transformations:
    - resource:
        kind: ConfigMap           # Required: resource kind
        group: ""                # Optional: API group (empty for core
resources)
        version: ""              # Optional: API version
        name: ""                  # Optional: specific resource name
    operations:
      - op: replace               # Operation type
        path: "/data/key"        # JSON Pointer path
        value: "new-value"        # New value (for add/replace/test)

```

Unterstützte Ressourcen

Sie können Ressourcentransformationen mit Ressourcen verwenden, die den folgenden Kriterien entsprechen:

- **Art** (erforderlich): Die Art der Kubernetes-Ressource (zum Beispiel ConfigMap, Deployment, Pod)
- **Gruppe** (optional): Die API-Gruppe (z. B. apps, route.openshift.io), für Kernressourcen weglassen
- **Version** (optional): Die API-Version (zum Beispiel v1, v1beta1)
- **Name** (optional): Gilt nur für eine bestimmte Ressource anhand des Namens



Die Modifizierung von PersistentVolumeClaims und Namespaces wird derzeit nicht unterstützt.

Unterstützte Operationen

Die folgenden Operationen können zur Änderung von Ressourcen verwendet werden: add, copy, move, remove, replace und test.

Einer Ressource einen Wert hinzufügen

Verwenden Sie die add-Operation, um ein neues Feld oder einen neuen Wert am angegebenen Pfad hinzuzufügen. Sie können Daten zu Objekten oder Arrays hinzufügen.

Verwenden Sie einen CR

Das folgende Beispiel fügt einer `Deployment` Ressource einen Knotenselektor hinzu:

```
transformations:
- resource:
  kind: Deployment
  operations:
  - op: add
    path: "/spec/template/spec/nodeSelector"
    value:
      "topology.kubernetes.io/zone": "us-east-1a"
      disktype: "ssd"
```

Verwenden Sie die Befehlszeile

Verwenden Sie den folgenden Befehl, um diese Transformation über die Befehlszeile auszuführen:

```
tridentctl-protect create snapshotrestore <restore_name> \
--snapshot <namespace/snapshot-name> \
--namespace-mapping <source-ns>:<dest-ns> \
--transformation
'apps,v1,Deployment:add{"path":"/spec/template/spec/nodeSelector","value":{"topology.kubernetes.io/zone":"us-east-1a","disktype":"ssd"}}'
```

Einen Wert innerhalb einer Ressource kopieren

Verwenden Sie die `copy` Operation, um einen Wert innerhalb derselben Ressource von einem Pfad in einen anderen zu kopieren. Die Quelle bleibt unverändert.

Verwenden Sie einen CR

Das folgende Beispiel dupliziert einen Datenschlüssel für ein ConfigMap Objekt:

```
transformations:
- resource:
  kind: ConfigMap
  operations:
  - op: copy
    from: "/data/source-key"
    path: "/data/backup-key"
```

Verwenden Sie die Befehlszeile

Verwenden Sie den folgenden Befehl, um diese Transformation über die Befehlszeile auszuführen:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation ',v1,ConfigMap:copy{"from":"/data/source-
key","path":"/data/backup-key"}'
```

Verschieben Sie einen Wert innerhalb einer Ressource

Verwenden Sie die `move`-Operation, um einen Wert innerhalb derselben Ressource von einem Pfad zu einem anderen zu verschieben. Die Quelle wird entfernt und der Wert am Zielort eingefügt.

Verwenden Sie einen CR

Das folgende Beispiel benennt einen Datenschlüssel für ein `ConfigMap` Objekt um:

```
transformations:
- resource:
  kind: ConfigMap
  operations:
  - op: move
    from: "/data/OLD_KEY"
    path: "/data/NEW_KEY"
```

Verwenden Sie die Befehlszeile

Verwenden Sie den folgenden Befehl, um diese Transformation über die Befehlszeile auszuführen:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:move{"from":"/data/OLD_KEY","path":"/data/NEW_KEY"}'
```

Einen Wert aus einer Ressource entfernen

Verwenden Sie die `remove` Operation, um ein Feld oder einen Wert am angegebenen Pfad zu entfernen.

Verwenden Sie einen CR

Das folgende Beispiel entfernt eine Annotation von einer `ConfigMap` Ressource:

```
transformations:
- resource:
```

```

    kind: ConfigMap
  operations:
    - op: remove
      path: "/metadata/annotations/kubectl.kubernetes.io~1last-
        applied-configuration"

```



Im Pfad des obigen Beispiels ~1 ist die JSON Pointer Escape-Sequenz für /.

Verwenden Sie die Befehlszeile

Verwenden Sie den folgenden Befehl, um diese Transformation über die Befehlszeile auszuführen:

```

tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
  ',v1,ConfigMap:remove{"path":"/metadata/annotations/kubectl.kubernetes.
  io~1last-applied-configuration"}'

```

Ersetzen Sie einen Wert innerhalb einer Ressource

Die replace-Operation ersetzt einen vorhandenen Wert innerhalb einer Ressource am angegebenen Pfad. Der JSON-Pfad muss bereits existieren.

Verwenden Sie einen CR

Das folgende Beispiel ändert den Hostnamen eines Route Objekts:

```

transformations:
  - resource:
      kind: Route
      group: route.openshift.io
    operations:
      - op: replace
        path: "/spec/host"
        value: "prod.example.com"

```

Verwenden Sie die Befehlszeile

Verwenden Sie den folgenden Befehl, um diese Transformation über die Befehlszeile auszuführen:

```

tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation

```

```
'route.openshift.io,v1,Route:replace{"path":"/spec/host","value":"prod.example.com"}'
```

Testen Sie die Ressourcenänderung

Die `test` Operation kann verwendet werden, um zu überprüfen, ob der Wert an einem Pfad dem erwarteten Wert entspricht, bevor die anderen Operationen angewendet werden. Wenn der Test fehlschlägt, wird die gesamte Änderung zurückgesetzt.

Verwenden Sie einen CR

Das folgende Beispiel aktualisiert `database-host` nur, wenn `environment` auf `staging` gesetzt ist:

```
transformations:
- resource:
  kind: ConfigMap
  operations:
  - op: test
    path: "/data/environment"
    value: "staging"
  - op: replace
    path: "/data/database-host"
    value: "prod-db.example.com"
```

Verwenden Sie die Befehlszeile

Verwenden Sie den folgenden Befehl, um diese Transformation über die Befehlszeile auszuführen:

```
tridentctl-protect create snapshotrestore <restore_name> \
--snapshot <namespace/snapshot-name> \
--namespace-mapping <source-ns>:<dest-ns> \
--transformation
',v1,ConfigMap:test{"path":"/data/environment","value":"staging"},repla
ce{"path":"/data/database-host","value":"prod-db.example.com"}'
```

Anwendungen mit NetApp SnapMirror und Trident Protect replizieren

Mit Trident Protect können Sie die asynchronen Replizierungsfunktionen der NetApp SnapMirror-Technologie nutzen, um Daten und Anwendungsänderungen von einem Storage-Backend auf ein anderes zu replizieren, entweder im selben Cluster oder zwischen verschiedenen Clustern.

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

Während Wiederherstellungs- und Failover-Vorgängen werden die Labels und Annotationen im Ziel-Namensraum so angepasst, dass sie den Labels und Annotationen im Quell-Namensraum entsprechen. Labels oder Annotationen aus dem Quell-Namensraum, die im Ziel-Namensraum nicht existieren, werden hinzugefügt, und alle Labels oder Annotationen, die bereits vorhanden sind, 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-Annotationen stellen sicher, dass wiederhergestellte Pods die entsprechenden Berechtigungen und Sicherheitskonfigurationen einhalten, die durch OpenShift Security Context Constraints (SCCs) definiert sind, und ohne Berechtigungsprobleme auf Volumes zugreifen können. Weitere Informationen finden sich in der ["OpenShift security context constraints Dokumentation"](#).

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

```
helm upgrade trident-protect -n trident-protect netapp-trident-protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_key_to_skip_2>}" \
  --reuse-values
```



Bei der Durchführung einer Wiederherstellungs- oder Failover-Operation werden alle in `restoreSkipNamespaceAnnotations` und `restoreSkipNamespaceLabels` angegebenen Namespace-Annotationen und -Labels von der Wiederherstellungs- oder Failover-Operation ausgeschlossen. Stellen Sie sicher, dass diese Einstellungen während der initialen Helm-Installation konfiguriert sind. Weitere Informationen sind unter ["Konfigurieren Sie zusätzliche Trident Protect Helm-Chart-Einstellungen"](#) zu finden.

Wenn Sie die Quellanwendung mit Helm mit dem `--create-namespace` Flag installiert haben, wird dem `name` Label-Schlüssel eine besondere Behandlung zuteil. Während des Wiederherstellungs- oder Failover-Prozesses kopiert Trident Protect dieses Label in den Ziel-Namespace, aktualisiert jedoch den Wert auf den Wert des Ziel-Namespace, wenn der Wert aus der Quelle mit dem Quell-Namespace übereinstimmt. Wenn dieser Wert nicht mit dem Quell-Namespace übereinstimmt, wird er unverändert in den Ziel-Namespace kopiert.

Beispiel

Das folgende Beispiel zeigt einen Quell- und einen Ziel-Namensraum mit jeweils unterschiedlichen Annotationen und Labels. Sie können den Zustand des Ziel-Namensraums vor und nach der Operation sehen 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-Namespace vor der Wiederherstellungs- oder Failover-Operation:

Namensraum	Anmerkungen	Etiketten
Namespace ns-1 (Quelle)	<ul style="list-style-type: none">• annotation.one/key: "updatedvalue"• annotation.two/key: "true"	<ul style="list-style-type: none">• environment=production• compliance=hipaa• name=ns-1
Namespace ns-2 (Ziel)	<ul style="list-style-type: none">• annotation.one/key: "true"• annotation.three/key: "false"	<ul style="list-style-type: none">• role=database

Nach dem Wiederherstellungsvorgang

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

Namensraum	Anmerkungen	Etiketten
Namespace ns-2 (Ziel)	<ul style="list-style-type: none">• annotation.one/key: "updatedvalue"• annotation.two/key: "true"• annotation.three/key: "false"	<ul style="list-style-type: none">• name=ns-2• compliance=hipaa• environment=production• role=database



Sie können Trident Protect so konfigurieren, dass Dateisysteme während Datensicherungsoperationen eingefroren und wieder freigegeben werden. ["Erfahren Sie mehr über die Konfiguration des Einfrierens von Dateisystemen mit Trident Protect"](#).

Ausführungshooks während Failover- und Reverse-Operationen

Wenn Sie eine AppMirror-Beziehung zum Schutz Ihrer Anwendung verwenden, gibt es bestimmte Verhaltensweisen im Zusammenhang mit Ausführungs-Hooks, die Sie während Failover- und Rückwärtsoperationen beachten sollten.

- Während des Failovers werden die Ausführungshooks automatisch vom Quell-Cluster auf den Ziel-Cluster kopiert. Sie müssen sie nicht manuell neu erstellen. Nach dem Failover sind die Ausführungshooks in der Anwendung vorhanden und werden bei allen relevanten Aktionen ausgeführt.
- Während der Reverse- oder Reverse-Resync werden alle vorhandenen Ausführungs-Hooks der Anwendung entfernt. Wenn die Quellanwendung zur Zielanwendung wird, sind diese Ausführungs-Hooks nicht mehr gültig und werden gelöscht, um ihre Ausführung zu verhindern.

Weitere Informationen zu Execution Hooks finden Sie unter ["Trident Protect-Ausführungshooks verwalten"](#).

Eine Replikationsbeziehung einrichten

Die Einrichtung einer Replikationsbeziehung umfasst Folgendes:

- Auswahl, wie häufig 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 Snapshots

Schritte

1. Erstellen Sie auf dem Quell-Cluster ein AppVault für die Quellenanwendung. Passen Sie je nach Speicheranbieter ein Beispiel in "[AppVault benutzerdefinierte Ressourcen](#)" an Ihre Umgebung an:

Erstellen Sie ein AppVault mit einem CR

- a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) 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 Custom Resource. Notieren Sie sich den Namen, den Sie wählen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diesen Wert verweisen.
 - **spec.providerConfig:** (*Erforderlich*) Speichert die Konfiguration, die für den Zugriff auf AppVault mit dem angegebenen Provider erforderlich ist. Wählen Sie einen `bucketName` und alle weiteren erforderlichen Details für Ihren Provider. Notieren Sie sich die Werte, die Sie wählen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diese Werte verweisen. Siehe "[AppVault benutzerdefinierte Ressourcen](#)" für Beispiele von AppVault CRs mit anderen Providern.
 - **spec.providerCredentials:** (*Erforderlich*) Speichert Verweise auf alle Anmeldeinformationen, die für den Zugriff auf den AppVault mit dem angegebenen Anbieter erforderlich sind.
 - **spec.providerCredentials.valueFromSecret:** (*Erforderlich*) Gibt an, dass der Anmeldeinformationswert aus einem Geheimnis stammen sollte.
 - **Schlüssel:** (*Erforderlich*) Der gültige Schlüssel des Geheimnisses, der ausgewählt werden soll.
 - **Name:** (*Erforderlich*) Name des Geheimnisses, das den Wert für dieses Feld enthält. Muss sich im selben Namespace befinden.
 - **spec.providerCredentials.secretAccessKey:** (*Erforderlich*) Der Zugriffsschlüssel, der für den Zugriff auf den Provider verwendet wird. Der **name** muss mit **spec.providerCredentials.valueFromSecret.name** übereinstimmen.
 - **spec.providerType:** (*Erforderlich*) Legt fest, was die Datensicherung bereitstellt; beispielsweise NetApp ONTAP S3, generisches S3, Google Cloud oder Microsoft Azure. Mögliche Werte:
 - `aws`
 - `azure`
 - `gcp`
 - `generic-s3`
 - `ontap-s3`
 - `storagegrid-s3`
- c. Nachdem die `trident-protect-appvault-primary-source.yaml` Datei mit den korrekten Werten gefüllt wurde, wird die CR angewendet:

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

Erstellen Sie ein AppVault mit der CLI

- a. Erstellen Sie die AppVault, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen:

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

2. Erstellen Sie auf dem Quell-Cluster die Quellenanwendungs-CR:

Erstellen Sie die Quellenanwendung mithilfe eines CR

- a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) 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 Namen, den Sie wählen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diesen Wert verweisen.
 - **spec.includedNamespaces:** (*Erforderlich*) Ein Array von Namespaces und zugehörigen Labels. Verwenden Sie Namespace-Namen und grenzen Sie optional den Geltungsbereich der Namespaces mit Labels ein, um Ressourcen anzugeben, die in den hier aufgeführten Namespaces vorhanden sind. Der Anwendungsnamespace 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 die `trident-protect-app-source.yaml` Datei mit den korrekten Werten gefüllt wurde, wird die CR angewendet:

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

namespace

Erstellen Sie die Quellanwendung mithilfe der CLI

a. Erstellen Sie die Quellanwendung. Beispiel:

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

3. Optional können Sie auf dem Quell-Cluster einen Snapshot der Quellanwendung erstellen. Dieser Snapshot dient als Grundlage für die Anwendung auf dem Ziel-Cluster. Wenn Sie diesen Schritt überspringen, müssen Sie warten, bis der nächste geplante Snapshot ausgeführt wird, damit Sie einen aktuellen Snapshot haben. Um einen bedarfsgesteuerten Snapshot zu erstellen, siehe ["Erstellen Sie einen On-Demand-Snapshot"](#).
4. Erstellen Sie auf dem Quell-Cluster den Replikationszeitplan CR:

Zusätzlich zum unten angegebenen Zeitplan wird empfohlen, einen separaten Zeitplan für tägliche Snapshots mit einer Aufbewahrungsdauer von 7 Tagen zu erstellen, um einen gemeinsamen Snapshot zwischen verbundenen ONTAP Clustern zu gewährleisten. Dadurch ist sichergestellt, dass Snapshots bis zu 7 Tage lang verfügbar sind, aber die Aufbewahrungsdauer kann basierend auf den Benutzeranforderungen angepasst werden.



Im Falle eines Failovers kann das System diese Snapshots bis zu 7 Tage lang für Rückoperationen nutzen. Dieser Ansatz macht den Rückprozess schneller und effizienter, da nur die seit dem letzten Snapshot vorgenommenen Änderungen übertragen werden, nicht 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) 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` der AppVault 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 der Wert muss auf 0 gesetzt werden.
 - **spec.enabled:** Muss auf `true` gesetzt werden.
 - **spec.granularity:** Muss auf `Custom` gesetzt werden.

- **spec.recurrenceRule:** Definieren Sie ein Startdatum in UTC-Zeit und ein Wiederholungsintervall.
- **spec.snapshotRetention:** Muss auf 2 eingestellt sein.

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"
```

- Nachdem die `trident-protect-schedule.yaml` Datei mit den korrekten Werten gefüllt wurde, wird die CR angewendet:

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

Erstellen Sie den Replikationszeitplan mithilfe der CLI

- 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 Ziel-Cluster eine Quellenanwendungs-AppVault-CR, die mit der AppVault-CR identisch ist, die Sie auf dem Quell-Cluster angewendet haben, 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 eine Ziel-AppVault CR für die Zielanwendung im Ziel-Cluster. Passen Sie je nach Speicheranbieter ein Beispiel in "[AppVault benutzerdefinierte Ressourcen](#)" an Ihre Umgebung an:
 - a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) 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 Custom Resource. Notieren Sie sich den Namen, den Sie wählen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diesen Wert verweisen.
 - **spec.providerConfig:** (*Erforderlich*) Speichert die Konfiguration, die für den Zugriff auf AppVault mit dem angegebenen Provider erforderlich ist. Wählen Sie einen `bucketName` und alle weiteren erforderlichen Details für Ihren Provider. Notieren Sie sich die Werte, die Sie wählen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diese Werte verweisen. Siehe "[AppVault benutzerdefinierte Ressourcen](#)" für Beispiele von AppVault CRs mit anderen Providern.
 - **spec.providerCredentials:** (*Erforderlich*) Speichert Verweise auf alle Anmeldeinformationen, die für den Zugriff auf den AppVault mit dem angegebenen Anbieter erforderlich sind.
 - **spec.providerCredentials.valueFromSecret:** (*Erforderlich*) Gibt an, dass der Anmeldeinformationswert aus einem Geheimnis stammen sollte.
 - **Schlüssel:** (*Erforderlich*) Der gültige Schlüssel des Geheimnisses, der ausgewählt werden soll.
 - **Name:** (*Erforderlich*) Name des Geheimnisses, das den Wert für dieses Feld enthält. Muss sich im selben Namespace befinden.
 - **spec.providerCredentials.secretAccessKey:** (*Erforderlich*) Der Zugriffsschlüssel, der für den Zugriff auf den Provider verwendet wird. Der **name** muss mit **spec.providerCredentials.valueFromSecret.name** übereinstimmen.
 - **spec.providerType:** (*Erforderlich*) Legt fest, was die Datensicherung bereitstellt; beispielsweise NetApp ONTAP S3, generisches S3, Google Cloud oder Microsoft Azure. Mögliche Werte:
 - `aws`
 - `azure`
 - `gcp`
 - `generic-s3`
 - `ontap-s3`
 - `storagegrid-s3`

- c. Nachdem die `trident-protect-appvault-secondary-destination.yaml` Datei mit den korrekten Werten gefüllt wurde, wird die CR angewendet:

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

8. Erstellen Sie auf dem Ziel-Cluster eine AppMirrorRelationship CR-Datei.



Bei Verwendung einer CR muss der Ziel-Namespace vor der Anwendung der CR manuell erstellt werden. Trident Protect erstellt Namespaces automatisch nur bei Verwendung der CLI.

Erstellen Sie eine AppMirrorRelationship mit einem CR

- Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (zum Beispiel `trident-protect-relationship.yaml`).
- Konfigurieren Sie die folgenden Attribute:
 - **metadata.name:** (Erforderlich) Der Name der AppMirrorRelationship benutzerdefinierten Ressource.
 - **spec.destinationAppVaultRef:** (Erforderlich) Dieser Wert muss mit dem Namen der AppVault für die Zieldanwendung auf dem Ziel-Cluster übereinstimmen.
 - **spec.namespaceMapping:** (Erforderlich) Die Ziel- und Quell-Namespace müssen mit dem in der jeweiligen Anwendungs-CR definierten Anwendungs-Namespace übereinstimmen.
 - **spec.sourceAppVaultRef:** (Erforderlich) Dieser Wert muss mit dem Namen der AppVault 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 auf dem Cluster. Die Speicherklasse muss mit einer ONTAP Storage-VM verknüpft sein, die mit der Quellumgebung gepaart ist. Wenn keine Speicherklasse angegeben wird, wird standardmäßig die Standard-Speicherklasse auf dem Cluster verwendet.
 - **spec.storageClassMapping:** (Optional) Ordnet Quellspeicherklassen Zielspeicherklassen für einzelne PVCs innerhalb desselben AMR zu. Jeder Eintrag erfordert einen `source` und `destination` Namen für die Speicherklasse. Wenn festgelegt, `storageClassMapping` überschreibt `storageClassName` vollständig; PVCs ohne übereinstimmenden Quelleintrag verwenden die Standardspeicherklasse des Clusters, nicht `storageClassName`.

Bei der Verwendung von `storageClassMapping` in einem AppMirrorRelationship CR sind die folgenden Einschränkungen zu beachten:

- `storageClassMapping` überschreibt `storageClassName` vollständig. Wenn beide Felder gesetzt sind, `storageClassName` wird ignoriert.
- PVCs mit Quellspeicherklassen, die keinen passenden Eintrag in `storageClassMapping` haben, verwenden die Standardspeicherklasse des Clusters,

nicht `storageClassName`.

- Jeder Eintrag erfordert sowohl ein `source` als auch `destination` Wert.
- Durch die Änderung der Zielspeicherklasse kann das Löschen und Neuerstellen vorhandener Ziel-PVCs erzwungen werden.
- **`spec.recurrenceRule`:** Definieren Sie ein Startdatum in UTC-Zeit und ein Wiederholungsintervall.

Beispiel YAML:

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

- c. Nachdem die `trident-protect-relationship.yaml` Datei mit den korrekten Werten gefüllt wurde, wird die CR angewendet:

```
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, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen:

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirrorrelationship> --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> --storageclass-mapping
<src_class:dst_class> -n <application_namespace>
```



Verwenden Sie `--storageclass-mapping` zur Zuordnung einzelner Quell-Speicherklassen zu Ziel-Speicherklassen. Ein oder mehrere source:destination-Paare, durch Kommas getrennt (zum Beispiel "class-a:class-b,class-c:class-d"), können angegeben werden. Verwenden Sie *:destination-class als Platzhalter, um alle nicht zugeordneten PVCs einer einzigen Ziel-Speicherklasse zuzuordnen. Wenn `--storageclass-mapping` festgelegt ist, wird `--storage-class` ignoriert.

Beispiel mit einer einzelnen Zielspeicherklasse (`--storage-class`):

```
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 Ziel-Cluster den Status und den Zustand der Replikationsbeziehung:

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

Failover zum Ziel-Cluster

Mit Trident Protect können Sie replizierte Anwendungen auf einen Ziel-Cluster umschalten. Dieses Verfahren stoppt die Replikationsbeziehung und bringt die Anwendung im Ziel-Cluster online. Trident Protect stoppt die Anwendung im Quell-Cluster nicht, wenn sie dort betriebsbereit war.

Schritte

1. Bearbeiten Sie auf dem Ziel-Cluster die AppMirrorRelationship CR-Datei (zum Beispiel `trident-protect-relationship.yaml`) und ändern Sie den Wert von **spec.desiredState** zu 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 übernommene Anwendung.
5. (Optional) Überprüfen Sie den Status und Zustand 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 eines Resynchronisierungsvorgangs wird die ursprüngliche Quellanwendung zur aktiven Anwendung und alle Änderungen, die an der aktiven Anwendung im Ziel-Cluster vorgenommen wurden, werden verworfen.

Der Prozess stoppt die App auf dem Ziel-Cluster, 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 Quell-Cluster einen Snapshot der Quellanwendung. Dadurch wird sichergestellt, dass die neuesten Änderungen vom Quell-Cluster erfasst werden.
2. Bearbeiten Sie auf dem Ziel-Cluster die AppMirrorRelationship CR-Datei (zum Beispiel `trident-protect-relationship.yaml`) und ändern Sie den Wert von `spec.desiredState` zu `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. Wenn Sie auf dem Ziel-Cluster Schutzzeitpläne für die ausgefallene Anwendung erstellt haben, entfernen Sie diese. Alle verbleibenden Zeitpläne verursachen Fehler bei Volume-Snapshots.

Reverse resync einer fehlgeschlagenen Replikationsbeziehung

Wenn Sie eine fehlgeschlagene Replikationsbeziehung rücksynchronisieren, wird die Zielanwendung zur Quellanwendung und die Quellanwendung zur Zielanwendung. Änderungen, die während des Failovers an der Zielanwendung vorgenommen wurden, bleiben erhalten.

Schritte

1. Löschen Sie auf dem ursprünglichen Ziel-Cluster die AppMirrorRelationship CR. Dadurch wird das Ziel-Cluster zum Quell-Cluster. Wenn auf dem neuen Ziel-Cluster noch Datensicherungsstrategien 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 gegenüberliegenden Cluster anwenden.
3. Stellen Sie sicher, dass das neue Ziel (ursprünglicher Quell-Cluster) mit beiden AppVault CRs konfiguriert ist.
4. Richten Sie eine Replikationsbeziehung auf dem gegenüberliegenden Cluster ein, und konfigurieren Sie Werte für die umgekehrte Richtung.

Replikationsrichtung der Anwendung umkehren

Wenn Sie die Replikationsrichtung umkehren, verschiebt Trident Protect die Anwendung zum Ziel-Storage-Backend und repliziert gleichzeitig weiterhin zurück zum ursprünglichen Quell-Storage-Backend. Trident Protect stoppt die Quellanwendung und repliziert die Daten zum Ziel, bevor auf die Zielanwendung umgeschaltet wird.

In dieser Situation tauschen Sie den Quell-Cluster und den Ziel-Cluster.

Schritte

1. Erstellen Sie auf dem Quell-Cluster einen Shutdown-Snapshot:

Erstellen Sie einen Shutdown-Snapshot mithilfe eines CR

- a. Deaktivieren Sie die Zeitpläne der Datensicherungsstrategie für die Quellanwendung.
- b. Erstellen Sie eine ShutdownSnapshot CR-Datei:
 - i. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) 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` der AppVault für die Quellanwendung übereinstimmen.
 - **spec.ApplicationRef:** *(Erforderlich)* Dieser Wert muss mit dem Feld `metadata.name` der Quellanwendungs-CR-Datei 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 die `trident-protect-shutdownsnapshot.yaml` Datei mit den korrekten Werten gefüllt wurde, wird die CR angewendet:

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

Erstellen Sie einen Shutdown-Snapshot mit der CLI

- a. Erstellen Sie den Shutdown-Snapshot, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Beispiel:

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

2. Auf dem Quell-Cluster, nachdem der Shutdown-Snapshot abgeschlossen ist, rufen Sie den Status des Shutdown-Snapshots ab:

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

3. Ermitteln Sie auf dem Quell-Cluster 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 Ziel-Cluster zum neuen Quell-Cluster mit der folgenden Änderung durch:



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

5. Führen Sie die umgekehrten Resynchronisierungsschritte in [Reverse resync einer fehlgeschlagenen Replikationsbeziehung](#) aus.
6. Aktivieren Sie Schutzzeitpläne auf dem neuen Quell-Cluster.

Ergebnis

Die folgenden Aktionen erfolgen aufgrund der umgekehrten Replikation:

- Es wird eine Snapshot 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 für 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 von Applikationen auf den ursprünglichen Quell-Cluster

Mit Trident Protect können Sie „Failback“ nach einem Failover-Vorgang durchführen, indem Sie die folgende Abfolge von Schritten ausführen. 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 übergewechselten Zustand.
- Reverse resync die Replikationsbeziehung.



Führen Sie keine normale Resynchronisierungsoperation durch, da dadurch Daten, die während des Failover-Vorgangs auf den Ziel-Cluster geschrieben wurden, verworfen werden.

- Die Replikationsrichtung umkehren.

Schritte

1. Führen Sie die [Reverse resync einer fehlgeschlagenen Replikationsbeziehung](#) Schritte aus.
2. Führen Sie die [Replikationsrichtung der Anwendung umkehren](#) Schritte aus.

Eine Replikationsbeziehung löschen

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

Schritte

1. Löschen Sie im aktuellen Ziel-Cluster 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 auf verschiedene Storage-Klassen migrieren, indem Sie Sicherungsdaten wiederherstellen.



Bei der Migration einer Anwendung werden alle für die Anwendung konfigurierten Ausführungs-Hooks mit der Anwendung migriert. Wenn ein Ausführungs-Hook nach der Wiederherstellung vorhanden ist, wird er automatisch als Teil des Wiederherstellungsvorgangs ausgeführt.

Sicherungs- und Wiederherstellungsvorgänge

Um Sicherungs- und Wiederherstellungsvorgänge für die folgenden Szenarien durchzuführen, können Sie

bestimmte Sicherungs- und Wiederherstellungsaufgaben automatisieren.

Klon auf denselben Cluster

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

Schritte

1. Führen Sie einen der folgenden Schritte aus:
 - a. ["Erstellen Sie einen Snapshot"](#).
 - b. ["Erstelle eine Sicherungskopie"](#).
2. Führen Sie auf demselben Cluster einen der folgenden Schritte aus, je nachdem, ob Sie einen Snapshot oder ein Backup 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 Quell-Cluster und stellen Sie diese Sicherung anschließend auf dem Ziel-Cluster wieder her. Stellen Sie sicher, dass Trident Protect auf dem Ziel-Cluster installiert ist.



Sie können eine Anwendung zwischen verschiedenen Clustern replizieren, indem Sie ["SnapMirror-Replikation"](#).

Schritte

1. ["Erstelle eine Sicherungskopie"](#).
2. Stellen Sie sicher, dass der AppVault CR für den Objektspeicher-Bucket, der die Sicherung enthält, auf dem Ziel-Cluster konfiguriert wurde.
3. Auf dem Ziel-Cluster ["Stellen Sie Ihre Daten aus der Sicherung wieder her"](#).

Anwendungen von einer Speicherklasse zu einer anderen Speicherklasse migrieren

Sie können Anwendungen von einer Speicherklasse in eine andere Speicherklasse migrieren, indem Sie ein Backup in der Ziel-Speicherklasse 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) 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 sinnvollen Namen für Ihre Umgebung.
 - **spec.appArchivePath:** Der Pfad innerhalb von AppVault, in dem die Snapshot-Inhalte gespeichert sind. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

```

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

```

- **spec.appVaultRef:** (*Erforderlich*) Der Name des AppVault, in dem die Snapshot-Inhalte gespeichert sind.
- **spec.namespaceMapping:** Die Zuordnung des Quell-Namespace des Wiederherstellungsvorgangs zum Ziel-Namespace. Ersetzen Sie `my-source-namespace` und `my-destination-namespace` durch 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 müssen, Filter hinzufügen, die Ressourcen mit bestimmten Bezeichnungen ein- oder

ausschließen:

- **resourceFilter.resourceSelectionCriteria:** (Für die Filterung erforderlich) Verwenden Sie `include` or `exclude`, um eine in `resourceMatchers` definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden `resourceMatchers`-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:
 - **resourceFilter.resourceMatchers:** Ein Array von `resourceMatcher`-Objekten. Wenn mehrere Elemente in diesem Array definiert werden, erfolgt die Übereinstimmung als ODER-Verknüpfung, und die Felder innerhalb jedes Elements (`group`, `kind`, `version`) werden als UND-Verknüpfung betrachtet.
 - **resourceMatchers[].group:** (*Optional*) Gruppe der zu filternden Ressource.
 - **resourceMatchers[].kind:** (*Optional*) Art der zu filternden Ressource.
 - **resourceMatchers[].version:** (*Optional*) Version der zu filternden Ressource.
 - **resourceMatchers[].names:** (*Optional*) Namen im Kubernetes `metadata.name`-Feld der Ressource, die gefiltert werden soll.
 - **resourceMatchers[].namespaces:** (*Optional*) Namespaces im Kubernetes `metadata.name`-Feld der Ressource, die gefiltert werden soll.
 - **resourceMatchers[].labelSelectors:** (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld `name` der Ressource, wie definiert in der "[Kubernetes-Dokumentation](#)". 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 die `trident-protect-snapshot-restore-cr.yaml` Datei mit den korrekten Werten gefüllt wurde, wird die CR angewendet:

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

Stellen Sie den Snapshot mithilfe der CLI wieder her.

Schritte

1. Stellen Sie den Snapshot in einem anderen Namespace wieder her, wobei Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen.
 - Das `snapshot` Argument verwendet einen Namespace und einen Snapshot-Namen im Format `<namespace>/<name>`.
 - Das `namespace-mapping` Argument verwendet durch Doppelpunkte getrennte Namensräume, um Quell-Namensräume den korrekten Ziel-Namensräumen im Format `source1:dest1,source2:dest2` zuzuordnen.

Beispiel:

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

Schutz von KubeVirt virtuellen Maschinen mit Trident Protect

Trident Protect ermöglicht das Sichern und Wiederherstellen von KubeVirt virtuellen Maschinen (VMs), die auf OpenShift Virtualization ausgeführt werden. Es ist möglich, alle VMs in einem oder mehreren Namespaces gleichzeitig zu schützen oder mit dem `includedVirtualMachines` Feld gezielt bestimmte VMs nach Namen auszuwählen. Trident Protect sammelt während der Sicherung automatisch alle abhängigen VM-Ressourcen. Bei der Wiederherstellung kann entweder das gesamte Archiv oder einzelne VMs gezielt wiederhergestellt werden.

KubeVirt Schutzmodi

Trident Protect unterstützt zwei Möglichkeiten, um festzulegen, welche KubeVirt VMs geschützt werden sollen. Ein Modus wird bei der Definition einer Anwendung ausgewählt:

Modus	Feld	Beschreibung
Namespace-basiert	<code>includedNamespaces</code>	Alle übereinstimmenden Ressourcen in einem oder mehreren Namespaces werden geschützt. Ein <code>labelSelector</code> im Namespace-Eintrag steuert dynamisch, welche VMs einbezogen werden.
VM-bezogen	<code>includedVirtualMachines</code>	Bestimmte KubeVirt VMs anhand von Namespace und Name schützen. Dieser Modus eignet sich, wenn VMs über mehrere Namespaces verteilt sind oder nicht alle Namespace-Ressourcen geschützt werden sollen.

Sie müssen genau ein Feld (`includedNamespaces` oder `includedVirtualMachines` im `ApplicationSpec` angeben. Die gleichzeitige Verwendung beider in derselben Anwendung ist nicht möglich.

Beide Modi unterstützen zusätzliche Filter:

- `resourceFilter` verwenden, um bestimmte Ressourcen einzuschließen oder auszuschließen.
- Mit `includedClusterScopedResources` können kompatible, clusterweite Ressourcen wie `StorageClass` Objekte einbezogen werden.



Trident Protect kann die KubeVirt VM-Dateisysteme während Datensicherungsoperationen einfrieren und wieder auftauen, um Konsistenz sicherzustellen. Einzelheiten zur Konfiguration sind unter "[Schutz von Daten mit KubeVirt VMs](#)" zu finden.

Während der Sicherung und des Snapshots erfasste Ressourcen

Wenn eine Sicherung oder ein Snapshot für eine Anwendung erstellt wird, die KubeVirt VMs umfasst, sammelt Trident Protect die ausgewählten VM-Ressourcen und erkennt automatisch deren abhängige Ressourcen.

Das Sammlungsverhalten gilt in zwei Fällen:

- VM-bezogene Anwendungen, die `includedVirtualMachines` verwenden.
- Namespace-basierte Anwendungen, die KubeVirt `VirtualMachine` Ressourcen enthalten.

VM-bezogene Sammlung

Für VM-bezogene Anwendungen beginnt Trident Protect mit jeder `VirtualMachine` in `includedVirtualMachines` aufgeführten VM und ermittelt die abhängigen Ressourcen, die zur Wiederherstellung dieser VM benötigt werden.

Ressourcentyp	Umfang	Wie es entdeckt wird
<code>VirtualMachine</code>	Namespace-bezogen	Direkter Treffer aus der <code>includedVirtualMachines</code> Liste.
<code>VirtualMachineInstance</code>	Namespace-bezogen	Gleicher Namespace und Name wie die VM, wenn die VM ausgeführt wird.
<code>VirtualMachineInstanceType</code>	Namespace-bezogen	Referenziert von <code>specinstancetype</code> wenn <code>kind: VirtualMachineInstanceType</code> .
<code>VirtualMachineClusterInstanceType</code>	Cluster	Wird referenziert durch <code>specinstancetype</code> wenn <code>kind</code> nicht gesetzt ist oder <code>kind: VirtualMachineClusterInstanceType</code> .
<code>VirtualMachinePreference</code>	Namespace-bezogen	Referenziert von <code>spec.preference</code> wenn <code>kind: VirtualMachinePreference</code> .
<code>VirtualMachineClusterPreference</code>	Cluster	Wird referenziert durch <code>spec.preference</code> wenn <code>kind</code> nicht gesetzt ist oder <code>kind: VirtualMachineClusterPreference</code> .
<code>DataVolume</code>	Namespace-bezogen	Referenziert von <code>spec.template.spec.volumes[].dataVolume.name</code> .

Ressourcentyp	Umfang	Wie es entdeckt wird
PersistentVolumeClaim	Namespace-bezogen	Erkannt aus DataVolume-backing PVCs, die denselben Namen wie die DataVolume verwenden; direkte PVC-Referenzen in <code>persistentVolumeClaim.claimName</code> ; Speicherabbild-PVCs in <code>memoryDump.claimName</code> ; und ephemere PVCs in <code>ephemeral.persistentVolumeClaim.claimName</code> .
PersistentVolume	Cluster	Für jedes erfasste PVC wird dies aus dem zugrunde liegenden PV mithilfe von <code>spec.volumeName</code> ermittelt.
Secret	Namespace-bezogen	Ermittelt aus Volume-Secrets in <code>secret.secretName</code> ; containerDisk Image Pull Secrets; CloudInitNoCloud und CloudInitConfigDrive Benutzer- und Netzwerkdaten-Secret-Referenzen; Sysprep Secrets; KernelBoot Container Image Pull Secrets; und AccessCredentials SSH- oder Passwort-Secrets.
ConfigMap	Namespace-bezogen	Entdeckt aus dem Volume ConfigMaps in <code>configMap.name</code> und Sysprep ConfigMaps.
ServiceAccount	Namespace-bezogen	Aus <code>`serviceAccount.serviceAccountName`</code> Volume-Referenzen ermittelt.
NetworkAttachmentDefinition	Namespace-bezogen	Ermittelt aus Multus-Netzwerken <code>spec.template.spec.networks[].multus.networkName</code> , wenn der Netzwerkname / nicht enthält. Es werden nur NetworkAttachmentDefinition-Verweise innerhalb desselben Namensraums erfasst. Verweise zwischen verschiedenen Namensräumen werden übersprungen.
Pod	Namespace-bezogen	Für <code>virt-launcher`</code> Pods durch Verwendung des Label-Selectors <code>`kubevirt.io=virt-launcher,vm.kubevirt.io/name=<vmName>`</code> erkannt.
Storage Class, Trident Volumes und Trident Backends	Cluster	Gesammelt auf die gleiche Weise wie bei namespace-basierten Anwendungen, basierend auf den für die VM ermittelten PVCs.



Nachdem alle VM-bezogenen Namespaced-Ressourcen erfasst wurden, wendet Trident Protect die Standard `resourceFilter`-Methode an, sofern eine konfiguriert ist. Die Erfassung von Cluster-bezogenen Ressourcen erfolgt anschließend wie gewohnt.

Namespace-basierte Sammlung

Für Namespace-basierte Anwendungen scannt Trident Protect die ausgewählten Namespaces und erfasst alle übereinstimmenden Ressourcen. Wenn KubeVirt `VirtualMachine` Objekte gefunden werden, sammelt Trident Protect automatisch deren abhängige Ressourcen, einschließlich Instanztypen, Präferenzen und ``virt-launcher``Pods, wie es auch bei VM-basierten Anwendungen geschieht.

Dies bedeutet, dass VM-spezifische Ressourcen auch dann in Ihre Sicherung oder Ihren Snapshot einbezogen werden, wenn die Anwendung mit `includedNamespaces` definiert ist.

VMs mit namespace-basierten Anwendungsdefinitionen schützen

Wenn eine namespace-basierte Anwendung erstellt wird, scannt Trident Protect die ausgewählten Namespaces und erfasst alle Ressourcen, einschließlich aller KubeVirt VMs. Für jede gefundene virtuelle

Maschine sammelt Trident Protect außerdem die abhängigen Ressourcen, die für das Sichern und Wiederherstellen dieser VM benötigt werden.

Beispiel für eine namensraumbasierte Anwendungs-CR:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedNamespaces:
    - namespace: <vm_namespace>
```

Nachdem Sie die Anwendung erstellt haben, werden die Standard Trident Protect Workflows für Backups und Snapshots verwendet:

- Beispiele für Snapshot YAML sind unter ["Erstellen Sie einen On-Demand-Snapshot"](#) zu finden.
- Beispiele für Backup-YAML-Dateien sind unter ["Erstellen Sie eine bedarfsgesteuerte Datensicherung"](#) zu finden.

Wiederherstellungsverhalten für namespace-basierte VM-Anwendungen

Da namespace-basierte Anwendungen Ressourcen aus den ausgewählten Namespaces sammeln, kann das Backup oder der Snapshot KubeVirt-Laufzeitobjekte und von KubeVirt verwaltete Metadaten enthalten. Einige dieser Objekte sollten nicht direkt wiederhergestellt werden, da sie den Laufzeitstatus des Quell-Cluster repräsentieren.

Während der Wiederherstellung wendet Trident Protect KubeVirt-aware-Transformationen an, um Quell-Cluster-Metadaten zu entfernen und temporäre KubeVirt-Ressourcen zu überspringen. Dies unterstützt KubeVirt dabei, die VM in der wiederhergestellten Umgebung sauber neu zu erstellen.

Ressource	Transformieren
VirtualMachine	Entfernt Beschriftungen und Anmerkungen, die mit <code>kubevirt.io/</code> beginnen.
VirtualMachineInstance	Entfernt Beschriftungen und Anmerkungen, die mit <code>kubevirt.io/</code> beginnen.
VirtualMachineInstanceMigration	Ignoriert und nicht wiederhergestellt.
VirtualMachineSnapshot	Ignoriert und nicht wiederhergestellt.
VirtualMachineSnapshotContent	Ignoriert und nicht wiederhergestellt.
Pod (virt-launcher)	Wird übersprungen, wenn das Pod das <code>kubevirt.io=virt-launcher</code> Label hat.



Trident Protect stellt KubeVirt-Ressourcen in einer definierten Reihenfolge wieder her, um Abhängigkeitsprobleme zu vermeiden. CDI DataVolumes (`cdi.kubevirt.io`) werden zuerst wiederhergestellt, damit die VM-Festplattendaten verfügbar sind, bevor die VM-Objekte erstellt

werden. Anschließend werden VirtualMachines (kubevirt.io vor VirtualMachineInstances wiederhergestellt.

Dynamischer VM-Schutz mit Label-Selektoren

Wenn Sie VMs dynamisch schützen möchten, ohne jeden VM-Namen aufzulisten, verwenden Sie namespace-basierte Anwendungsdefinitionen mit Label-Selektoren.

Nur die VirtualMachine CR benötigt Labels. Abhängige Ressourcen wie DataVolumes, PVCs, Secrets und ConfigMaps werden automatisch erfasst.

Schritte

1. Virtuelle Maschinen-CRs mit einem gemeinsamen Label versehen:

```
kubectl label virtualmachine <vm_name_1> app=<label_value> -n  
<vm_namespace>  
kubectl label virtualmachine <vm_name_2> app=<label_value> -n  
<vm_namespace>
```

2. Eine Anwendungs-CR mit includedNamespaces und einem labelSelector erstellen:

```
apiVersion: protect.trident.netapp.io/v1  
kind: Application  
metadata:  
  name: <application_name>  
  namespace: <application_namespace>  
spec:  
  includedNamespaces:  
    - namespace: <vm_namespace>  
    labelSelector:  
      matchLabels:  
        app: <label_value>
```

3. Backups und Wiederherstellungen werden mit dem Standard-Trident Protect Workflow erstellt.

Weitere Details:

- VMs können jederzeit durch Ändern der Bezeichnungen zum Schutz hinzugefügt oder daraus entfernt werden:

```
kubectl label virtualmachine <vm_name_to_add> app=<label_value> -n  
<vm_namespace>  
kubectl label virtualmachine <vm_name_to_remove> app  
=<different_label_value> -n <vm_namespace>
```

- Sie können Label-Selektoren auch über mehrere Namensräume hinweg verwenden:

```
spec:
  includedNamespaces:
    - namespace: <vm_namespace_1>
      labelSelector:
        matchLabels:
          <label_key>: <label_value>
    - namespace: <vm_namespace_2>
      labelSelector:
        matchLabels:
          <label_key>: <label_value>
```

Bestimmte VMs mit includedVirtualMachines schützen

Verwenden Sie `includedVirtualMachines` diese Funktion, wenn eine benannte Gruppe von `VirtualMachines` über einen oder mehrere Namespaces hinweg geschützt und bestimmte VMs selektiv aus einem Archiv wiederhergestellt werden sollen.

Sowohl die Anwendungs-CRs als auch die Wiederherstellungs-CRs verwenden das gleiche `includedVirtualMachines` Selector-Format.

Dieser Selektor kann in verschiedenen Kontexten verwendet werden:

- In einer Anwendungs-CR ist festgelegt, welche VMs geschützt sind.
- In einem Wiederherstellungs-CR wird definiert, welche VMs aus dem Backup- oder Snapshot-Archiv wiederhergestellt werden.



Bei der Wiederherstellung von CRs `includedVirtualMachines` ist die Angabe optional. Wenn dieses Feld nicht enthalten ist, wird das gesamte Archiv wiederhergestellt. Wenn es angegeben ist, werden nur die aufgeführten VMs und ihre abhängigen Ressourcen wiederhergestellt.

Das folgende YAML zeigt die gemeinsame Selektorstruktur:

```
includedVirtualMachines:
  - namespace: <vm_namespace_1>
    names:
      - <vm_name_1>
      - <vm_name_2>
  - namespace: <vm_namespace_2>
    names:
      - <vm_name_3>
```



Mehrere Einträge in `includedVirtualMachines` können auf unterschiedliche Namensräume verweisen, was die Definition und Wiederherstellung von VM-fähigen Anwendungen mit

mehreren Namensräumen ermöglicht.

Feld	Typ	Beschreibung
Namensraum	Zeichenkette	Kubernetes Namespace, der die Ziel-VM-Objekte enthält.
Namen	Liste von Zeichenketten	Ein oder mehrere virtuelle Maschinennamen in diesem Namensraum.

Beispiel einer VM-basierten Anwendung:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedVirtualMachines:
    - namespace: <vm_namespace_1>
      names:
        - <vm_name_1>
        - <vm_name_2>
    - namespace: <vm_namespace_2>
      names:
        - <vm_name_3>
  includedClusterScopedResources:
    - group: storage.k8s.io
      kind: StorageClass
```

Bestimmte KubeVirt VMs wiederherstellen

Sie können `includedVirtualMachines` auf Wiederherstellungs-CRs anwenden, um nur ausgewählte KubeVirt VMs aus einem Backup, Snapshot oder replizierten Snapshot-Archiv wiederherzustellen.

Wenn `includedVirtualMachines` in einem Wiederherstellungs-CR angegeben ist, stellt Trident Protect nur die aufgeführten `VirtualMachines` und deren abhängige Ressourcen wie `VirtualMachineInstance`, `DataVolume`, `PersistentVolumeClaim`, `PersistentVolume`, `Secret`, `ConfigMap`, `ServiceAccount` und andere VM-bezogene Objekte wieder her.

Wenn `includedVirtualMachines` im Restore-CR nicht angegeben wird, stellt Trident Protect das vollständige Archiv entsprechend der Restore-CR-Konfiguration wieder her.

Unterstützte Wiederherstellungs-CRs und CLI-Befehle

Die folgende Tabelle listet die Wiederherstellungs-CRs auf, die `includedVirtualMachines` unterstützen, sowie die entsprechenden CLI-Befehle.

CR-Wiederherstellung	CLI Befehl	Beschreibung
BackupRestore	<code>tridentctl-protect create backuprestore</code>	Stellt Daten aus einer Sicherung in neue oder andere Namespace-Zuordnungen wieder her. Unterstützt <code>namespaceMapping</code> und <code>storageClassMapping</code> .
BackupInplaceRestore	<code>tridentctl-protect create backupinplacerestore</code>	Stellt eine Sicherung im ursprünglichen Namensraum oder in den ursprünglichen Namensräumen wieder her.
SnapshotRestore	<code>tridentctl-protect create snapshotrestore</code>	Stellt aus einem Snapshot neue oder andere Namespace-Zuordnungen wieder her. Unterstützt <code>namespaceMapping</code> und <code>storageClassMapping</code> .
SnapshotInplaceRestore	<code>tridentctl-protect create snapshotinplace restore</code>	Stellt einen Snapshot im ursprünglichen Namensraum oder in den ursprünglichen Namensräumen wieder her.
ReplicateSnapshotRestore	Nur YAML oder den unterstützten CLI-Befehl verwenden, falls in Ihrer Umgebung verfügbar	Stellt Daten aus einem replizierten Snapshot wieder her. Unterstützt <code>namespaceMapping</code> , <code>storageClassMapping</code> und Wiederherstellung vor Ort mithilfe des <code>inPlaceRestore</code> Flags.



Die `tridentctl-protect` CLI bietet keine Option für `includedVirtualMachines` bei Wiederherstellungsbefehlen. Um bestimmte VMs wiederherzustellen, muss eine Wiederherstellungs-CR-YAML-Datei direkt mit `kubectl apply` angewendet werden. Die YAML-Datei kann von Grund auf neu erstellt oder mit der CLI unter Verwendung von `--dry-run --output yml` eine Startdatei generiert werden. Anschließend wird das `includedVirtualMachines` Feld unter `spec` hinzugefügt und die Datei angewendet.

VM-bezogenes Wiederherstellungsverhalten

Das `includedVirtualMachines` Feld in einem Wiederherstellungs-CR verwendet dasselbe Selektorformat wie das Anwendungs-CR:

```
includedVirtualMachines:
- namespace: <source_vm_namespace>
  names:
    - <vm_name_1>
    - <vm_name_2>
```

Bei Wiederherstellungs-CRs beziehen sich die Namespace-Werte in `includedVirtualMachines` auf die Quell-Namespace, wie sie im Backup, Snapshot oder replizierten Snapshot-Archiv vorliegen.

Wenn Sie `namespaceMapping` Trident verwenden, wählt Trident Protect zunächst die angeforderten VMs aus dem Quellarchiv aus und wendet anschließend die Namespace-Zuordnungen während der Wiederherstellung an. Wenn Sie `storageClassMapping` Trident verwenden, wendet Trident Protect die Storage-Class-

Zuordnungen während der Transformationsphase der Wiederherstellung an.

Wiederherstellungs-YAML mit der Befehlszeile generieren

Mit `tridentctl-protect` lässt sich eine Basis-Restore-CR generieren, anschließend kann die YAML-Datei bearbeitet werden, um `includedVirtualMachines` hinzuzufügen.

```
tridentctl-protect create <restore_command> <restore_name> \
  <restore_options> \
  --dry-run --output yaml > <restore_name>.yaml
```

Bearbeiten Sie die generierte YAML-Datei und fügen Sie das `includedVirtualMachines` Feld unter `spec` hinzu, dann die Datei anwenden:

```
kubectl apply -f <restore_name>.yaml
```

Beispiel: Definition einer VM-basierten Anwendung

Die folgende Anwendung schützt drei spezifische VMs in zwei Namensräumen.

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: production-vms
  namespace: prod
spec:
  includedVirtualMachines:
    - namespace: prod
      names:
        - app-server
        - database-server
    - namespace: monitoring
      names:
        - prometheus-vm
  includedClusterScopedResources:
    - group: storage.k8s.io
      kind: StorageClass
```

Beispiel: Backup erstellen

Für die Erstellung einer Sicherung einer VM-basierten Anwendung wird die Standard-Backup-CR verwendet. VM-spezifische Felder sind in der Backup-CR nicht erforderlich.

```
apiVersion: protect.trident.netapp.io/v1
```

```

kind: Backup
metadata:
  name: daily-backup
  namespace: prod
spec:
  applicationRef: production-vms
  appVaultRef: s3-vault

```

Beispiel: Wiederherstellung einer einzelnen VM aus einer Sicherung

Das folgende Beispiel stellt nur database-server aus dem Backup-Archiv wieder her. Die anderen VMs im Archiv, wie app-server und prometheus-vm, werden übersprungen.

```

apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: restore-db-only
  namespace: prod-vms-restore
spec:
  appVaultRef: s3-vault
  appArchivePath: backups/production-vms/2026-03-10T00-00-00Z
  namespaceMapping:
    - source: prod
      destination: prod-dr
  includedVirtualMachines:
    - namespace: prod
      names:
        - database-server

```

```
kubectl apply -f restore-db-only.yaml
```

Nur database-server und die davon abhängigen Ressourcen wie DataVolumes, PVCs, PVs, Secrets, ConfigMaps und zugehörige KubeVirt-Ressourcen werden wiederhergestellt.

Beispiel: Wiederherstellung einer einzelnen VM vor Ort aus einer Sicherung

Das folgende Beispiel stellt nur database-server in seinen ursprünglichen Namensraum aus dem Sicherungsarchiv wieder her.

```

apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: ipr-db-only
  namespace: prod

```

```
spec:
  appVaultRef: s3-vault
  appArchivePath: backups/production-vms/2026-03-10T00-00-00Z
  includedVirtualMachines:
    - namespace: prod
      names:
        - database-server
```

```
kubectl apply -f ipr-db-only.yaml
```

Beispiel: Wiederherstellung einer einzelnen VM aus einem Snapshot

Das folgende Beispiel stellt nur ubuntu-vm-blue-a627be38 aus einem Snapshot-Archiv wieder her.

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: blue-vm-snap-restore
  namespace: blue
spec:
  appVaultRef: my-appvault
  appArchivePath: snapshots/protectctl-blue-vm/snap-1
  includedVirtualMachines:
    - namespace: blue
      names:
        - ubuntu-vm-blue-a627be38
```

```
kubectl apply -f blue-vm-snap-restore.yaml
```

VM-Wiederherstellung mit anderen Filtern kombinieren

Sie können `includedVirtualMachines` zusammen mit `resourceFilter`, `namespaceMapping` und `storageClassMapping` im selben Wiederherstellungs-CR verwenden, um genau festzulegen, was wiederhergestellt wird.

Wenn eine `resourceFilter` hinzugefügt wird, wählt Trident Protect zunächst die in `includedVirtualMachines` aufgeführten VMs und deren abhängige Ressourcen aus. Anschließend wird der Ressourcenfilter angewendet, um die ein- oder auszuschließenden Ressourcen aus diesen Ergebnissen weiter einzugrenzen.

```
spec:
  includedVirtualMachines:
    - namespace: <vm_namespace>
```

```

names:
  - <vm_name>
resourceFilter:
  resourceSelectionCriteria: Include
  resourceMatchers:
    - kinds: ["PersistentVolumeClaim"]
      names: ["<pvc_name>"]

```

Für BackupRestore, SnapshotRestore und ReplicateSnapshotRestore, namespaceMapping und storageClassMapping **arbeiten zusammen** mit includedVirtualMachines. Namespace-Referenzen in includedVirtualMachines entsprechen den Quell-Namespace, wie sie im Archiv erscheinen. Namespace- und Speicherklassenzuordnungen werden während der Wiederherstellungstransformationsphase nach Abschluss der Filterung angewendet.

Einzelheiten zum Wiederherstellungsverhalten sind unter ["Anwendungen mit Trident Protect wiederherstellen"](#) zu finden.

Eine VM-bezogene Anwendung mithilfe der Befehlszeilenschnittstelle erstellen

Das `--virtual-machines`-Flag kann verwendet werden, um eine Anwendung auf bestimmte KubeVirt VMs anstatt auf ganze Namensräume zu beschränken. Das `--virtual-machines`-Flag schließt sich mit `--namespaces` gegenseitig aus.

Schritte

1. Die Anwendung wird erstellt, indem einer der folgenden Befehle ausgeführt und die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzt werden. Das Format für das `--virtual-machines` Flag ist `<namespace> (<vm_name>)`. Bei mehreren VMs werden die Namen in den Klammern durch Kommas getrennt angegeben:

- Einzelne VM in einem Namespace:

```

tridentctl-protect create application <application_name> \
  -n <application_namespace> \
  --virtual-machines "<vm_namespace>(<vm_name>)"

```

- Mehrere VMs in verschiedenen Namensräumen:

```

tridentctl-protect create application <application_name> \
  -n <application_namespace> \
  --virtual-machines "<vm_namespace_1>(<vm_name_1>,<vm_name_2>
),<vm_namespace_2>(<vm_name_3>)"

```

Verwandte Informationen

- ["Eine Anwendung für das Management definieren"](#)
- ["Anwendungen schützen"](#)

- ["Anwendungen mit Trident Protect wiederherstellen"](#)
- ["Erweiterte Wiederherstellungseinstellungen"](#)
- ["Installieren Sie das Trident Protect CLI-Plugin"](#)
- ["Verwenden Sie AppVault-Objekte, um Buckets zu verwalten"](#)

Trident Protect-Ausführungshooks verwalten

Ein Ausführungshook ist eine benutzerdefinierte Aktion, die Sie so konfigurieren können, dass sie in Verbindung mit einem Datenschutzvorgang einer verwalteten App ausgeführt wird. Wenn Sie beispielsweise eine Datenbank-App haben, können Sie einen Ausführungshook verwenden, um alle Datenbanktransaktionen vor einem Snapshot anzuhalten und die Transaktionen nach Abschluss des Snapshots fortzusetzen. Dies gewährleistet applikationskonsistente Snapshots.

Arten von Execution Hooks

Trident Protect unterstützt die folgenden Arten von Ausführungshooks, basierend darauf, wann sie ausgeführt werden können:

- Vor-Snapshot
- Nach dem Schnappschuss
- Pre-Backup
- Nach dem Backup
- Nach der Wiederherstellung
- Nach dem Failover

Ausführungsreihenfolge

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

1. Alle anwendbaren benutzerdefinierten Pre-Operation-Hooks werden in den entsprechenden Containern ausgeführt. Sie können beliebig viele benutzerdefinierte Pre-Operation-Hooks erstellen und ausführen, aber die Ausführungsreihenfolge dieser Hooks vor der Operation ist weder garantiert noch konfigurierbar.
2. Gegebenenfalls kommt es zu Dateisystem-Freezes. ["Erfahren Sie mehr über die Konfiguration des Einfrierens von Dateisystemen mit Trident Protect"](#).
3. Der Datenschutzvorgang wird durchgeführt.
4. Eingefrorene Dateisysteme werden, falls zutreffend, wieder freigegeben.
5. Alle anwendbaren benutzerdefinierten Nachbearbeitungs-Hooks werden in den entsprechenden Containern ausgeführt. Sie können beliebig viele benutzerdefinierte Nachbearbeitungs-Hooks erstellen und ausführen, aber die Ausführungsreihenfolge 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 Ausführungsreihenfolge von Hooks unterschiedlicher Typen ist jedoch garantiert. Zum Beispiel ist dies die Ausführungsreihenfolge einer Konfiguration, die alle verschiedenen Hook-Typen enthält:

1. Pre-Snapshot-Hooks ausgeführt
2. Post-Snapshot-Hooks ausgeführt
3. Pre-Backup-Hooks ausgeführt
4. Post-Backup-Hooks ausgeführt



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



Sie sollten Ihre Ausführungshook-Skripte immer testen, bevor Sie sie in einer Produktionsumgebung aktivieren. Sie können den Befehl 'kubectl exec' verwenden, um die Skripte bequem zu testen. Nachdem Sie die Ausführungshooks 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 das Backup und in jede nachfolgende Wiederherstellungsoperation einbezogen.

Wichtige Hinweise zu benutzerdefinierten Ausführungshooks

Beachten Sie Folgendes bei der Planung von Execution Hooks für Ihre Apps.

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



Da Ausführungshooks die Funktionalität der Anwendung, auf der sie ausgeführt werden, oft einschränken oder vollständig deaktivieren, sollten Sie stets versuchen, die Zeit zu minimieren, die Ihre benutzerdefinierten Ausführungshooks zum Ausführen benötigen. Wenn Sie einen Sicherungs- oder Snapshot-Vorgang mit zugehörigen Ausführungshooks starten, diesen aber abbrechen, dürfen die Hooks dennoch ausgeführt werden, falls der Sicherungs- oder Snapshot-Vorgang bereits begonnen hat. Das bedeutet, dass die Logik, die in einem Ausführungshook nach der Sicherung verwendet wird, nicht davon ausgehen kann, dass die Sicherung abgeschlossen wurde.

Ausführungs-Hook-Filter

Wenn Sie einen Ausführungshook für eine Anwendung hinzufügen oder bearbeiten, können Sie Filter zum Ausführungshook hinzufügen, um zu steuern, auf welche Container der Hook angewendet wird. Filter sind nützlich für Anwendungen, die auf allen Containern dasselbe Container-Image verwenden, aber jedes Image für einen anderen Zweck nutzen (wie beispielsweise Elasticsearch). Filter ermöglichen die Erstellung von Szenarien, in denen Ausführungshooks auf einigen, aber nicht unbedingt allen identischen Containern ausgeführt werden. Wenn mehrere Filter für einen einzelnen Ausführungshook erstellt werden, werden sie mit einem logischen UND-Operator kombiniert. Pro Ausführungshook können bis zu 10 Filter aktiv sein.

Jeder Filter, den Sie einem Ausführungshook hinzufügen, verwendet einen regulären Ausdruck, um Container in Ihrem Cluster zu finden. Wenn ein Hook mit einem Container übereinstimmt, führt der Hook das zugehörige Skript auf diesem Container aus. Reguläre Ausdrücke für Filter verwenden die Regular Expression 2 (RE2) Syntax, die das Erstellen eines Filters, der Container von der Liste der Treffer ausschließt, nicht unterstützt. Weitere Informationen zur Syntax, die Trident Protect für reguläre Ausdrücke in Ausführungshook-Filtern unterstützt, finden Sie unter "[Unterstützung der Regular Expression 2 \(RE2\) Syntax](#)".



Wenn Sie einem Ausführungshook, der nach einem Wiederherstellungs- oder Klonvorgang ausgeführt wird, einen Namespace-Filter hinzufügen und sich die Quelle und das Ziel des Wiederherstellungs- oder Klonvorgangs in unterschiedlichen Namespaces befinden, wird der Namespace-Filter nur auf den Ziel-Namespace angewendet.

Beispiele für Execution Hooks

Besuchen Sie das "[NetApp Verda GitHub-Projekt](#)", um echte Ausführungs-Hooks für beliebte Apps wie Apache Cassandra und Elasticsearch herunterzuladen. Sie können auch Beispiele sehen und Ideen für die Strukturierung Ihrer eigenen benutzerdefinierten Ausführungs-Hooks erhalten.



Die Skripte im Verda-Repository werden wie besehen bereitgestellt und sind nicht offiziell von NetApp unterstützt. NetApp-Support umfasst ausschließlich das Execution Hooks Framework innerhalb von Trident Protect.

Erstellen Sie einen Ausführungs-Hook

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

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-hook.yaml`.
2. Konfigurieren Sie die folgenden Attribute so, dass sie Ihrer Trident Protect-Umgebung und Clusterkonfiguration entsprechen:
 - **metadata.name:** (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen 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
 - Beitrag
 - **spec.action:** (*Erforderlich*) Eine Zeichenkette, die angibt, welche Aktion der Ausführungs-Hook ausführt, vorausgesetzt, dass alle angegebenen Ausführungs-Hook-Filter übereinstimmen. Mögliche Werte:
 - Schnappschuss
 - Backup

- Wiederherstellen
- Failover
- **spec.enabled:** (*Optional*) Gibt an, ob dieser Ausführungs-Hook aktiviert oder deaktiviert ist. Wenn nicht angegeben, 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 und der Standardwert beträgt 25 Minuten, falls nicht angegeben.
- **spec.arguments:** (*Optional*) Eine YAML-Liste von Argumenten, die Sie für den Ausführungshook angeben können.
- **spec.matchingCriteria:** (*Optional*) Eine optionale Liste von Kriterien-Schlüssel-Wert-Paaren, wobei jedes Paar einen Ausführungs-Hook-Filter bildet. Sie können bis zu 10 Filter pro Ausführungs-Hook hinzufügen.
- **spec.matchingCriteria.type:** (*Optional*) Eine Zeichenkette, die den Filtertyp des Ausführungshooks 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 Sie die Befehlszeile

Schritte

1. Erstellen Sie den Ausführungs-Hook, und ersetzen Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung. Beispiel:

```

tridentctl-protect create exehook <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 Ausführungshook manuell ausführen

Sie können einen Ausführungs-Hook manuell zu Testzwecken ausführen oder wenn Sie den Hook nach einem Fehler manuell erneut ausführen müssen. Sie benötigen die Berechtigungen „Besitzer“, „Administrator“ oder „Mitglied“, um Ausführungs-Hooks manuell auszuführen.

Das manuelle Ausführen eines Execution Hooks besteht aus zwei grundlegenden Schritten:

1. Erstellen Sie eine Ressourcensicherung, die Ressourcen sammelt und eine Sicherung davon erstellt, wobei festgelegt wird, wo der Hook ausgeführt wird
2. Führe den Ausführungshook gegen das Backup aus

▼ Schritt 1: Erstellen Sie eine Ressourcensicherung

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-resource-backup.yaml`.
2. Konfigurieren Sie die folgenden Attribute so, dass sie Ihrer Trident Protect-Umgebung und Clusterkonfiguration entsprechen:
 - **metadata.name:** *(Erforderlich)* Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen 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 AppVault, in dem die Sicherungsinhalte gespeichert sind.
 - **spec.appArchivePath:** Der Pfad innerhalb von AppVault, in dem die Sicherungsinhalte gespeichert sind. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

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

Beispiel YAML:

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

3. Nachdem Sie die CR-Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

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

Verwenden Sie die Befehlszeile

Schritte

1. Erstellen Sie die Sicherungskopie, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. 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. Zeigen Sie den Status der Datensicherung an. Sie können diesen Beispielbefehl wiederholt

verwenden, bis der Vorgang abgeschlossen ist:

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

3. Überprüfen Sie, ob das Backup 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) 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 sinnvollen 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 aus dem ResourceBackup CR übereinstimmt, den Sie in Schritt 1 erstellt haben.

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

- **spec.action:** (*Erforderlich*) Eine Zeichenkette, die angibt, welche Aktion der Ausführungs-Hook ausführt, vorausgesetzt, dass alle angegebenen Ausführungs-Hook-Filter übereinstimmen. Mögliche Werte:
 - Schnappschuss
 - Backup
 - Wiederherstellen
 - Failover
- **spec.stage:** (*Erforderlich*) Eine Zeichenkette, die angibt, in welcher Phase der Aktion der Ausführungs-Hook ausgeführt werden soll. Dieser Hook-Lauf führt keine Hooks in anderen Phasen aus. Mögliche Werte:
 - Vor

- Beitrag

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 Sie die Befehlszeile

Schritte

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

```
tridentctl protect create exehooksruntime <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 ausgeführten Hooks. Sie können diesen Befehl so lange wiederholen, bis der Vorgang abgeschlossen ist:

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

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

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

Copyright-Informationen

Copyright © 2026 NetApp. Alle Rechte vorbehalten. Gedruckt in den USA. Dieses urheberrechtlich geschützte Dokument darf ohne die vorherige schriftliche Genehmigung des Urheberrechtsinhabers in keiner Form und durch keine Mittel – weder grafische noch elektronische oder mechanische, einschließlich Fotokopieren, Aufnehmen oder Speichern in einem elektronischen Abrufsystem – auch nicht in Teilen, vervielfältigt werden.

Software, die von urheberrechtlich geschütztem NetApp Material abgeleitet wird, unterliegt der folgenden Lizenz und dem folgenden Haftungsausschluss:

DIE VORLIEGENDE SOFTWARE WIRD IN DER VORLIEGENDEN FORM VON NETAPP ZUR VERFÜGUNG GESTELLT, D. H. OHNE JEGLICHE EXPLIZITE ODER IMPLIZITE GEWÄHRLEISTUNG, EINSCHLIESSLICH, JEDOCH NICHT BESCHRÄNKT AUF DIE STILLSCHWEIGENDE GEWÄHRLEISTUNG DER MARKTGÄNGIGKEIT UND EIGNUNG FÜR EINEN BESTIMMTEN ZWECK, DIE HIERMIT AUSGESCHLOSSEN WERDEN. NETAPP ÜBERNIMMT KEINERLEI HAFTUNG FÜR DIREKTE, INDIREKTE, ZUFÄLLIGE, BESONDERE, BEISPIELHAFTE SCHÄDEN ODER FOLGESCHÄDEN (EINSCHLIESSLICH, JEDOCH NICHT BESCHRÄNKT AUF DIE BESCHAFFUNG VON ERSATZWAREN ODER -DIENSTLEISTUNGEN, NUTZUNGS-, DATEN- ODER GEWINNVERLUSTE ODER UNTERBRECHUNG DES GESCHÄFTSBETRIEBS), UNABHÄNGIG DAVON, WIE SIE VERURSACHT WURDEN UND AUF WELCHER HAFTUNGSTHEORIE SIE BERUHEN, OB AUS VERTRAGLICH FESTGELEGTER HAFTUNG, VERSCHULDENSUNABHÄNGIGER HAFTUNG ODER DELIKTSHAFTUNG (EINSCHLIESSLICH FAHRLÄSSIGKEIT ODER AUF ANDEREM WEGE), DIE IN IRGEND EINER WEISE AUS DER NUTZUNG DIESER SOFTWARE RESULTIEREN, SELBST WENN AUF DIE MÖGLICHKEIT DERARTIGER SCHÄDEN HINGEWIESEN WURDE.

NetApp behält sich das Recht vor, die hierin beschriebenen Produkte jederzeit und ohne Vorankündigung zu ändern. NetApp übernimmt keine Verantwortung oder Haftung, die sich aus der Verwendung der hier beschriebenen Produkte ergibt, es sei denn, NetApp hat dem ausdrücklich in schriftlicher Form zugestimmt. Die Verwendung oder der Erwerb dieses Produkts stellt keine Lizenzierung im Rahmen eines Patentrechts, Markenrechts oder eines anderen Rechts an geistigem Eigentum von NetApp dar.

Das in diesem Dokument beschriebene Produkt kann durch ein oder mehrere US-amerikanische Patente, ausländische Patente oder anhängige Patentanmeldungen geschützt sein.

ERLÄUTERUNG ZU „RESTRICTED RIGHTS“: Nutzung, Vervielfältigung oder Offenlegung durch die US-Regierung unterliegt den Einschränkungen gemäß Unterabschnitt (b)(3) der Klausel „Rights in Technical Data – Noncommercial Items“ in DFARS 252.227-7013 (Februar 2014) und FAR 52.227-19 (Dezember 2007).

Die hierin enthaltenen Daten beziehen sich auf ein kommerzielles Produkt und/oder einen kommerziellen Service (wie in FAR 2.101 definiert) und sind Eigentum von NetApp, Inc. Alle technischen Daten und die Computersoftware von NetApp, die unter diesem Vertrag bereitgestellt werden, sind gewerblicher Natur und wurden ausschließlich unter Verwendung privater Mittel entwickelt. Die US-Regierung besitzt eine nicht ausschließliche, nicht übertragbare, nicht unterlizenzierbare, weltweite, limitierte unwiderrufliche Lizenz zur Nutzung der Daten nur in Verbindung mit und zur Unterstützung des Vertrags der US-Regierung, unter dem die Daten bereitgestellt wurden. Sofern in den vorliegenden Bedingungen nicht anders angegeben, dürfen die Daten ohne vorherige schriftliche Genehmigung von NetApp, Inc. nicht verwendet, offengelegt, vervielfältigt, geändert, aufgeführt oder angezeigt werden. Die Lizenzrechte der US-Regierung für das US-Verteidigungsministerium sind auf die in DFARS-Klausel 252.227-7015(b) (Februar 2014) genannten Rechte beschränkt.

Markeninformationen

NETAPP, das NETAPP Logo und die unter <http://www.netapp.com/TM> aufgeführten Marken sind Marken von NetApp, Inc. Andere Firmen und Produktnamen können Marken der jeweiligen Eigentümer sein.