



Sicherung von Applikationen mit Trident-Schutz

Trident

NetApp
September 26, 2025

Inhalt

Sicherung von Applikationen mit Trident-Schutz	1
Weitere Informationen zu Trident Protect	1
Was kommt als Nächstes?	1
Installieren Sie Trident Protect	1
Trident Protect – die Anforderungen	1
Installieren und konfigurieren Sie Trident Protect	4
Installieren Sie das Trident Protect CLI-Plug-in	10
Management von Trident Protect	14
Managen von Trident Schützen Sie die Autorisierung und Zugriffssteuerung	14
Generieren Sie ein Trident Protect Supportpaket	20
Upgrade von Trident Protect	22
Managen und sichern Sie Applikationen	22
Verwenden Sie zum Verwalten von Buckets Trident Protect AppVault-Objekte	22
Definieren Sie mit Trident Protect eine Applikation für das Management	31
Sicherung von Applikationen mit Trident Protect	33
Stellen Sie Applikationen mit Trident Protect wieder her	41
Replizieren Sie Applikationen mit NetApp SnapMirror und Trident Protect	57
Migrieren Sie Applikationen mit Trident Protect	71
Managen von Trident Protect-Ausführungshaken	75
Deinstallieren Sie Trident Protect	80

Sicherung von Applikationen mit Trident-Schutz

Weitere Informationen zu Trident Protect

NetApp Trident Protect bietet erweiterte Funktionen für das Applikations-Datenmanagement, mit denen die Funktionalität und Verfügbarkeit zustandsbehafteter Kubernetes-Applikationen auf der Basis von NetApp ONTAP Storage-Systemen und der NetApp Trident CSI Storage-bereitstellung verbessert werden. Trident Protect vereinfacht das Management, die Sicherung und die Verschiebung von Container-Workloads zwischen Public Clouds und On-Premises-Umgebungen. Außerdem bietet es Automatisierungsfunktionen über seine API und CLI.

Sie können Anwendungen mit Trident Protect schützen, indem Sie benutzerdefinierte Ressourcen (CRS) erstellen oder die Trident Protect CLI verwenden.

Was kommt als Nächstes?

Informieren Sie sich vor der Installation über die Trident Protect-Anforderungen:

- ["Trident Protect – die Anforderungen"](#)

Installieren Sie Trident Protect

Trident Protect – die Anforderungen

Prüfen Sie zunächst die Bereitschaft Ihrer Betriebsumgebung, Applikations-Cluster, Applikationen und Lizenzen. Stellen Sie sicher, dass Ihre Umgebung diese Anforderungen für den Einsatz und Betrieb von Trident Protect erfüllt.

Trident schützen die Kompatibilität von Kubernetes-Clustern

Trident Protect ist mit einer Vielzahl vollständig gemanagter und selbst gemanagter Kubernetes-Angebote kompatibel, darunter:

- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- Microsoft Azure Kubernetes Service (AKS)
- Red hat OpenShift
- SUSE Rancher
- VMware Tanzu Portfolio
- Vorgelagerte Kubernetes-Systeme



Stellen Sie sicher, dass der Cluster, auf dem Sie Trident Protect installieren, mit einem laufenden Snapshot-Controller und den zugehörigen CRDs konfiguriert ist. Informationen zum Installieren eines Snapshot-Controllers finden Sie unter ["Diese Anweisungen"](#).

Trident schützen die Kompatibilität mit dem Storage-Back-End

Trident Protect unterstützt die folgenden Storage-Back-Ends:

- Amazon FSX für NetApp ONTAP
- Cloud Volumes ONTAP
- ONTAP Storage-Arrays
- Google Cloud NetApp Volumes
- Azure NetApp Dateien

Stellen Sie sicher, dass Ihr Storage-Back-End die folgenden Anforderungen erfüllt:

- Vergewissern Sie sich, dass mit dem Cluster verbundener NetApp-Storage Astra Trident 24.02 oder neuer verwendet (Trident 24.10 wird empfohlen).
 - Wenn Astra Trident älter als die Version 24.06.1 ist und Sie die Disaster-Recovery-Funktion von NetApp SnapMirror nutzen möchten, müssen Sie die Astra Control Provisioner manuell aktivieren.
- Stellen Sie sicher, dass Sie über die neueste Astra Control Provisioner-Version verfügen (standardmäßig installiert und aktiviert ab Astra Trident 24.06.1).
- Stellen Sie sicher, dass Sie über ein NetApp ONTAP Storage-Back-End verfügen.
- Stellen Sie sicher, dass Sie einen Objekt-Storage-Bucket zum Speichern von Backups konfiguriert haben.
- Erstellen Sie alle Anwendungsnamepaces, die Sie für Anwendungen oder Verwaltungsvorgänge von Anwendungsdaten verwenden möchten. Trident Protect erstellt diese Namespaces nicht für Sie. Wenn Sie einen nicht vorhandenen Namespace in einer benutzerdefinierten Ressource angeben, schlägt der Vorgang fehl.

Anforderungen für nas-Economy-Volumen

Trident Protect unterstützt Backup- und Restore-Vorgänge auf Volumes mit nas-Wirtschaft. Snapshots, Klone und SnapMirror-Replizierung auf nas-Economy Volumes werden derzeit nicht unterstützt. Sie müssen ein Snapshot-Verzeichnis für jedes nas-Economy-Volumen aktivieren, das Sie mit Trident Protect verwenden möchten.



Einige Anwendungen sind nicht mit Volumes kompatibel, die ein Snapshot-Verzeichnis verwenden. Für diese Anwendungen müssen Sie das Snapshot-Verzeichnis ausblenden, indem Sie den folgenden Befehl auf dem ONTAP-Speichersystem ausführen:

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

Sie können das Snapshot-Verzeichnis aktivieren, indem Sie den folgenden Befehl für jedes nas-Economy-Volume ausführen und durch die UUID des Volumes ersetzen <volume-UUID>, das Sie ändern möchten:

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level=true -n trident
```



Sie können Snapshot-Verzeichnisse standardmäßig für neue Volumes aktivieren, indem Sie die Trident-Backend-Konfigurationsoption auf `true` setzen `snapshotDir`. Vorhandene Volumes werden nicht beeinträchtigt.

Datensicherung mit KubeVirt VMs

Trident Protect 24.10 und 24.10.1 und neuere Versionen weisen beim Schutz von Applikationen auf KubeVirt VMs unterschiedliche Verhaltensweisen auf. Bei beiden Versionen können Sie das Einfrieren und Entsperrn des Dateisystems während der Datenschutzvorgänge aktivieren oder deaktivieren.



Für alle Trident Protect-Versionen müssen Sie möglicherweise dem Application Namespace privilegierte Berechtigungen erteilen, um die Funktion zum automatischen Einfrieren in OpenShift-Umgebungen zu aktivieren oder zu deaktivieren. Beispiel:

```
oc adm policy add-scc-to-user privileged -z default -n  
<application-namespace>
```

Trident Protect 24.10

Trident Protect 24.10 stellt nicht automatisch einen konsistenten Zustand für KubeVirt VM-Dateisysteme während Datensicherheitsvorgängen sicher. Wenn Sie Ihre KubeVirt VM-Daten mit Trident Protect 24.10 schützen möchten, müssen Sie die Freeze/Unfreeze-Funktion für die Dateisysteme vor dem Datenschutzvorgang manuell aktivieren. Dadurch wird sichergestellt, dass sich die Dateisysteme in einem konsistenten Zustand befinden.

Sie können Trident Protect 24.10 so konfigurieren, dass das Einfrieren und Entfrieren des VM-Dateisystems während der Datenschutzvorgänge verwaltet wird, indem ["Konfiguration der Virtualisierung"](#) Sie dann den folgenden Befehl verwenden:

```
kubectl set env deployment/trident-protect-controller-manager  
NEPTUNE_VM_FREEZE=true -n trident-protect
```

Trident Protect 24.10.1 und neuer

Ab Trident Protect 24.10.1 friert Trident Protect KubeVirt-Dateisysteme während der Datensicherheitsvorgänge automatisch ein und entfriert diese. Optional können Sie dieses automatische Verhalten mit dem folgenden Befehl deaktivieren:

```
kubectl set env deployment/trident-protect-controller-manager  
NEPTUNE_VM_FREEZE=false -n trident-protect
```

Anforderungen für die SnapMirror-Replizierung

NetApp SnapMirror steht zusammen mit Trident Protect für die folgenden ONTAP Lösungen zur Verfügung:

- NetApp ASA
- NetApp AFF
- NetApp FAS

- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSX für NetApp ONTAP

ONTAP-Cluster-Anforderungen für die SnapMirror-Replizierung

Stellen Sie sicher, dass Ihr ONTAP Cluster die folgenden Anforderungen erfüllt, wenn Sie SnapMirror Replizierung nutzen möchten:

- **Astra Control Provisioner oder Trident:** Astra Control Provisioner oder Trident müssen sowohl auf den Quell- als auch auf den Ziel-Kubernetes-Clustern vorhanden sein, die ONTAP als Backend verwenden. Trident Protect unterstützt die Replikation mit NetApp SnapMirror-Technologie unter Verwendung von Storage-Klassen, die von den folgenden Treibern gesichert werden:
 - `ontap-nas`
 - `ontap-san`
- **Lizenzen:** Asynchrone Lizenzen von ONTAP SnapMirror, die das Datensicherungspaket verwenden, müssen sowohl auf den Quell- als auch auf den Ziel-ONTAP-Clustern aktiviert sein. Weitere Informationen finden Sie unter ["Übersicht über die SnapMirror Lizenzierung in ONTAP"](#).

Peering-Überlegungen für die SnapMirror-Replizierung

Stellen Sie sicher, dass Ihre Umgebung die folgenden Anforderungen erfüllt, wenn Sie Storage-Back-End-Peering verwenden möchten:

- **Cluster und SVM:** Die ONTAP Speicher-Back-Ends müssen aktiviert werden. Weitere Informationen finden Sie unter ["Übersicht über Cluster- und SVM-Peering"](#).



Vergewissern Sie sich, dass die in der Replizierungsbeziehung zwischen zwei ONTAP-Clustern verwendeten SVM-Namen eindeutig sind.

- **Astra Control Provisioner oder Trident und SVM:** Die Remote-SVMs müssen für die Astra Control Bereitstellung oder die Trident im Ziel-Cluster verfügbar sein.
- **Managed Back-Ends:** Sie müssen ONTAP-Speicher-Back-Ends in Trident Protect hinzufügen und managen, um eine Replikationsbeziehung zu erstellen.
- **NVMe über TCP:** Trident Protect unterstützt keine NetApp SnapMirror-Replizierung für Storage-Back-Ends, die das NVMe-over-TCP-Protokoll verwenden.

Trident/ONTAP-Konfiguration für SnapMirror-Replikation

Trident Protect setzt voraus, dass Sie mindestens ein Storage-Back-End konfigurieren, das die Replizierung sowohl für die Quell- als auch für Ziel-Cluster unterstützt. Wenn die Quell- und Ziel-Cluster identisch sind, sollte die Zielanwendung ein anderes Speicher-Back-End als die Quellanwendung verwenden, um die beste Ausfallsicherheit zu erreichen.

Installieren und konfigurieren Sie Trident Protect

Wenn Ihre Umgebung die Anforderungen für Trident Protect erfüllt, können Sie mit den folgenden Schritten Trident Protect auf Ihrem Cluster installieren. Sie können Trident Protect von NetApp beziehen oder es von Ihrer eigenen privaten Registrierung installieren. Die Installation von einer privaten Registrierung ist hilfreich, wenn Ihr Cluster

nicht auf das Internet zugreifen kann.



Standardmäßig erfasst Trident Protect Support-Informationen, die bei allen NetApp-Supportfällen helfen, die Sie öffnen können, einschließlich Protokollen, Kennzahlen und Topologieinformationen zu Clustern und gemanagten Applikationen. Trident Protect sendet diese Support-Pakete nach einem täglichen Zeitplan an NetApp. Sie können diese Support-Bundle-Sammlung optional deaktivieren, wenn Sie Trident Protect installieren. Sie können jederzeit manuell ["Generieren Sie ein Support-Bundle"](#) wählen.

Installieren Sie Trident Protect

Installieren Sie Trident Protect von NetApp

Schritte

1. Trident Helm Repository hinzufügen:

```
helm repo add netapp-trident-protect  
https://netapp.github.io/trident-protect-helm-chart
```

2. Installieren Sie die Trident Protect CRDs:

```
helm install trident-protect-crds netapp-trident-protect/trident-  
protect-crds --version 100.2410.1 --create-namespace --namespace  
trident-protect
```

3. Verwenden Sie Helm, um Trident Protect mit einem der folgenden Befehle zu installieren. Ersetzen Sie `<name_of_cluster>` ihn durch einen Cluster-Namen, der dem Cluster zugewiesen wird und zum Identifizieren der Backups und Snapshots des Clusters verwendet wird:

- Trident Protect normal installieren:

```
helm install trident-protect netapp-trident-protect/trident-  
protect --set clusterName=<name_of_cluster> --version 100.2410.1  
--create-namespace --namespace trident-protect
```

- Installieren Sie Trident Protect und deaktivieren Sie die geplanten täglichen Uploads von Trident Protect AutoSupport Support-Paketen:

```
helm install trident-protect netapp-trident-protect/trident-  
protect --set autoSupport.enabled=false --set  
clusterName=<name_of_cluster> --version 100.2410.1 --create  
-namespace --namespace trident-protect
```

Installieren Sie Trident Protect aus einer privaten Registrierung

Sie können Trident Protect aus einer privaten Image-Registrierung installieren, wenn Ihr Kubernetes-Cluster nicht auf das Internet zugreifen kann. Ersetzen Sie in diesen Beispielen Werte in Klammern durch Informationen aus Ihrer Umgebung:

Schritte

1. Ziehen Sie die folgenden Bilder auf Ihren lokalen Computer, aktualisieren Sie die Tags und schieben Sie sie dann in Ihre private Registrierung:

```
netapp/controller:24.10.1
netapp/restic:24.10.1
netapp/kopia:24.10.1
netapp/trident-autosupport:24.10.0
netapp/execheek:24.10.1
netapp/resourcebackup:24.10.1
netapp/resourcerestore:24.10.1
netapp/resourcedelete:24.10.1
bitnami/kubectl:1.30.2
kubebuilder/kube-rbac-proxy:v0.16.0
```

Beispiel:

```
docker pull netapp/controller:24.10.1
```

```
docker tag netapp/controller:24.10.1 <private-registry-
url>/controller:24.10.1
```

```
docker push <private-registry-url>/controller:24.10.1
```

2. Trident Protect System Namespace erstellen:

```
kubectl create ns trident-protect
```

3. Melden Sie sich bei der Registrierung an:

```
helm registry login <private-registry-url> -u <account-id> -p <api-
token>
```

4. Erstellen Sie einen Pull-Schlüssel, der für die Authentifizierung der privaten Registrierung verwendet werden soll:

```
kubectl create secret docker-registry regcred --docker
-username=<registry-username> --docker-password=<api-token> -n
trident-protect --docker-server=<private-registry-url>
```

5. Trident Helm Repository hinzufügen:

```
helm repo add netapp-trident-protect
https://netapp.github.io/trident-protect-helm-chart
```

6. Erstellen Sie eine Datei mit dem Namen `protectValues.yaml`. Stellen Sie sicher, dass es die folgenden Trident Protect-Einstellungen enthält:

```
---
image:
  registry: <private-registry-url>
imagePullSecrets:
  - name: regcred
controller:
  image:
    registry: <private-registry-url>
rbacProxy:
  image:
    registry: <private-registry-url>
crCleanup:
  imagePullSecrets:
    - name: regcred
webhooksCleanup:
  imagePullSecrets:
    - name: regcred
```

7. Installieren Sie die Trident Protect CRDs:

```
helm install trident-protect-crds netapp-trident-protect/trident-
protect-crds --version 100.2410.1 --create-namespace --namespace
trident-protect
```

8. Verwenden Sie Helm, um Trident Protect mit einem der folgenden Befehle zu installieren. Ersetzen Sie `<name_of_cluster>` ihn durch einen Cluster-Namen, der dem Cluster zugewiesen wird und zum Identifizieren der Backups und Snapshots des Clusters verwendet wird:

- Trident Protect normal installieren:

```
helm install trident-protect netapp-trident-protect/trident-
protect --set clusterName=<name_of_cluster> --version 100.2410.1
--create-namespace --namespace trident-protect -f
protectValues.yaml
```

- Installieren Sie Trident Protect und deaktivieren Sie die geplanten täglichen Uploads von Trident Protect AutoSupport Support-Paketen:

```
helm install trident-protect netapp-trident-protect/trident-protect --set autoSupport.enabled=false --set clusterName=<name_of_cluster> --version 100.2410.1 --create --namespace --namespace trident-protect -f protectValues.yaml
```

Geben Sie die Höchstwerte für die Trident Protect Container-Ressourcen an

Nach der Installation von Trident Protect können Sie eine Konfigurationsdatei verwenden, um Ressourcenbeschränkungen für Trident Protect Container festzulegen. Durch das Festlegen der Grenzwerte für die Ressourcen können Sie steuern, wie viele Ressourcen des Clusters von Trident Protect-Vorgängen verbraucht werden.

Schritte

1. Erstellen Sie eine Datei mit dem Namen `resourceLimits.yaml`.
2. Füllen Sie die Datei entsprechend den Anforderungen Ihrer Umgebung mit Optionen für die Ressourcenbegrenzung für Trident Container.

Die folgende Beispielkonfigurationsdatei zeigt die verfügbaren Einstellungen und enthält die Standardvaules für jedes Ressourcenlimit:

```
---
jobResources:
  defaults:
    limits:
      cpu: 8000m
      memory: 10000Mi
      ephemeralStorage: ""
    requests:
      cpu: 100m
      memory: 100Mi
      ephemeralStorage: ""
  resticVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  resticVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

```

requests:
  cpu: ""
  memory: ""
  ephemeralStorage: ""
kopiaVolumeBackup:
  limits:
    cpu: ""
    memory: ""
    ephemeralStorage: ""
  requests:
    cpu: ""
    memory: ""
    ephemeralStorage: ""
kopiaVolumeRestore:
  limits:
    cpu: ""
    memory: ""
    ephemeralStorage: ""
  requests:
    cpu: ""
    memory: ""
    ephemeralStorage: ""

```

3. Anwenden der Werte aus der `resourceLimits.yaml` Datei:

```

helm upgrade trident-protect -n trident-protect -f <resourceLimits.yaml>
--reuse-values

```

Installieren Sie das Trident Protect CLI-Plug-in

Sie können das Befehlszeilenplugin Trident Protect verwenden, das eine Erweiterung des Dienstprogramms Trident ist `tridentctl`, um benutzerdefinierte Ressourcen (Trident Protect Custom Resources, CRS) zu erstellen und mit ihnen zu interagieren.

Installieren Sie das Trident Protect CLI-Plug-in

Bevor Sie das Befehlszeilendienstprogramm verwenden, müssen Sie es auf dem Computer installieren, mit dem Sie auf das Cluster zugreifen. Befolgen Sie diese Schritte, je nachdem, ob Ihr Computer eine x64- oder ARM-CPU verwendet.

Download Plugin für Linux AMD64 CPUs

Schritte

1. Laden Sie das Trident Protect CLI-Plug-in herunter:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-linux-amd64
```

Download Plugin für Linux ARM64 CPUs

Schritte

1. Laden Sie das Trident Protect CLI-Plug-in herunter:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-linux-arm64
```

Download Plugin für Mac AMD64 CPUs

Schritte

1. Laden Sie das Trident Protect CLI-Plug-in herunter:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-macos-amd64
```

Download Plugin für Mac ARM64 CPUs

Schritte

1. Laden Sie das Trident Protect CLI-Plug-in herunter:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-macos-arm64
```

1. Ausführungsberechtigungen für die Plug-in-Binärdatei aktivieren:

```
chmod +x tridentctl-protect
```

2. Kopieren Sie die Plugin-Binärdatei an einen Speicherort, der in Ihrer PFADVARIABLE definiert ist. Beispiel /usr/bin: Oder /usr/local/bin (Sie benötigen möglicherweise erhöhte Privileges):

```
cp ./tridentctl-protect /usr/local/bin/
```

3. Optional können Sie die Plugin-Binärdatei an einen Speicherort in Ihrem Home-Verzeichnis kopieren. In diesem Fall wird empfohlen, sicherzustellen, dass der Speicherort Teil Ihrer PFADVARIABLE ist:

```
cp ./tridentctl-protect ~/bin/
```



Das Kopieren des Plugins an einen Ort in Ihrer PFADVARIABLE ermöglicht es Ihnen, das Plugin durch Eingabe oder `tridentctl protect` von einem beliebigen Ort zu verwenden `tridentctl-protect`.

Trident CLI Plug-in-Hilfe ansehen

Sie können die integrierten Plug-in-Hilfefunktionen nutzen, um detaillierte Hilfe zu den Funktionen des Plug-ins zu erhalten:

Schritte

1. Verwenden Sie die Hilfefunktion, um die Verwendungshilfe anzuzeigen:

```
tridentctl-protect help
```

Aktivieren Sie die automatische Vervollständigung von Befehlen

Nachdem Sie das Trident Protect CLI-Plugin installiert haben, können Sie die automatische Vervollständigung für bestimmte Befehle aktivieren.

Aktivieren Sie die automatische Vervollständigung für die Bash-Shell

Schritte

1. Download des Abschlussskripts:

```
curl -L -O https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-completion.bash
```

2. Erstellen Sie ein neues Verzeichnis in Ihrem Home-Verzeichnis, um das Skript zu enthalten:

```
mkdir -p ~/.bash/completions
```

3. Das heruntergeladene Skript in das Verzeichnis verschieben ~/.bash/completions:

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. Fügen Sie der Datei im Home-Verzeichnis folgende Zeile hinzu ~/.bashrc:

```
source ~/.bash/completions/tridentctl-completion.bash
```

Aktivieren Sie die automatische Vervollständigung für die Z-Shell

Schritte

1. Download des Abschlussskripts:

```
curl -L -O https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-completion.zsh
```

2. Erstellen Sie ein neues Verzeichnis in Ihrem Home-Verzeichnis, um das Skript zu enthalten:

```
mkdir -p ~/.zsh/completions
```

3. Das heruntergeladene Skript in das Verzeichnis verschieben ~/.zsh/completions:

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. Fügen Sie der Datei im Home-Verzeichnis folgende Zeile hinzu ~/.zprofile:

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

Ergebnis

Bei Ihrer nächsten Shell-Anmeldung können Sie den Befehl Auto-Vervollständigung mit dem tridentctl-Protect-Plugin verwenden.

Management von Trident Protect

Managen von Trident Schützen Sie die Autorisierung und Zugriffssteuerung

Trident Protect nutzt das Kubernetes-Modell der rollenbasierten Zugriffssteuerung (Role Based Access Control, RBAC). Standardmäßig stellt Trident Protect einen einzelnen System-Namespace und sein dazugehöriges Standarddienstkonto bereit. Wenn Ihr Unternehmen über eine Vielzahl von Benutzern oder spezifische Sicherheitsanforderungen verfügt, können Sie die RBAC-Funktionen von Trident Protect verwenden, um eine granularere Kontrolle über den Zugriff auf Ressourcen und Namespaces zu erlangen.

Der Clusteradministrator hat immer Zugriff auf Ressourcen im Standard- `trident-protect` Namespace und kann auch auf Ressourcen in allen anderen Namespaces zugreifen. Um den Zugriff auf Ressourcen und Anwendungen zu kontrollieren, müssen Sie zusätzliche Namespaces erstellen und diesen Namespaces Ressourcen und Anwendungen hinzufügen.

Beachten Sie, dass keine Benutzer Anwendungsdatenmanagement-CRS im Standard-Namespace erstellen können `trident-protect`. Sie müssen Anwendungsdatenmanagement-CRS in einem Anwendungs-Namespace erstellen (als Best Practice erstellen Sie Anwendungsdatenmanagement-CRS im gleichen Namespace wie ihre zugehörige Anwendung).



Nur Administratoren sollten Zugriff auf privilegierte Trident haben, die benutzerdefinierte Ressourcenobjekte schützen, darunter:

- **AppVault:** Erfordert Bucket-Zugangsdaten
- **AutoSupportBundle:** Sammelt Kennzahlen, Protokolle und andere sensible Trident schützen Daten
- **AutoSupportBundleSchedule:** Verwaltet Zeitpläne für die Protokollsammlung

Verwenden Sie als Best Practice RBAC, um den Zugriff auf privilegierte Objekte auf Administratoren zu beschränken.

Weitere Informationen darüber, wie RBAC den Zugriff auf Ressourcen und Namespaces regelt, finden Sie im ["RBAC-Dokumentation für Kubernetes"](#).

Informationen zu Servicekonten finden Sie im ["Dokumentation des Kubernetes Service-Kontos"](#).

Beispiel: Zugriff für zwei Benutzergruppen verwalten

Ein Unternehmen verfügt beispielsweise über einen Cluster-Administrator, eine Gruppe von Engineering-Benutzern und eine Gruppe von Marketing-Benutzern. Der Clusteradministrator führt die folgenden Aufgaben aus, um eine Umgebung zu erstellen, in der die Engineering-Gruppe und die Marketing-Gruppe jeweils nur auf die Ressourcen zugreifen können, die ihren jeweiligen Namespaces zugewiesen sind.

Schritt 1: Erstellen Sie einen Namespace, der Ressourcen für jede Gruppe enthält

Durch das Erstellen eines Namespace können Sie Ressourcen logisch trennen und besser kontrollieren, wer Zugriff auf diese Ressourcen hat.

Schritte

1. Erstellen Sie einen Namespace für die Engineering-Gruppe:

```
kubectl create ns engineering-ns
```

2. Erstellen Sie einen Namespace für die Marketinggruppe:

```
kubectl create ns marketing-ns
```

Schritt 2: Erstellen Sie neue Dienstkonten, um mit Ressourcen in jedem Namespace zu interagieren

Jeder neue Namespace, den Sie erstellen, verfügt über ein Standard-Dienstkonto. Sie sollten jedoch für jede Benutzergruppe ein Dienstkonto erstellen, damit Sie Privileges bei Bedarf in Zukunft weiter zwischen Gruppen aufteilen können.

Schritte

1. Erstellen Sie ein Servicekonto für die Engineering-Gruppe:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. Erstellen Sie ein Service-Konto für die Marketinggruppe:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

Schritt 3: Erstellen Sie ein Geheimnis für jedes neue Service-Konto

Ein Dienstkontogeheimnis wird verwendet, um sich beim Dienstkonto zu authentifizieren. Er kann bei einer Kompromittierung einfach gelöscht und neu erstellt werden.

Schritte

1. Einen Schlüssel für das Engineering-Servicekonto erstellen:

```

apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: eng-user
  name: eng-user-secret
  namespace: engineering-ns
  type: kubernetes.io/service-account-token

```

2. Erstellen Sie ein Geheimnis für das Marketingservicekonto:

```

apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: mkt-user
  name: mkt-user-secret
  namespace: marketing-ns
  type: kubernetes.io/service-account-token

```

Schritt 4: Erstellen Sie ein RoleBinding-Objekt, um das ClusterRole-Objekt an jedes neue Servicekonto zu binden

Bei der Installation von Trident Protect wird ein Standardobjekt für ClusterRole erstellt. Sie können diese ClusterRole an das Dienstkonto binden, indem Sie ein RoleBinding-Objekt erstellen und anwenden.

Schritte

1. Binden Sie die ClusterRole an das Engineering-Servicekonto:

```

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: engineering-ns-tenant-rolebinding
  namespace: engineering-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns

```

2. Binden Sie den ClusterRole an das Marketingservicekonto:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: marketing-ns-tenant-rolebinding
  namespace: marketing-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: mkt-user
  namespace: marketing-ns
```

Schritt 5: Testberechtigungen

Überprüfen Sie, ob die Berechtigungen korrekt sind.

Schritte

1. Bestätigung, dass Engineering-Benutzer auf Engineering-Ressourcen zugreifen können:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n engineering-ns
```

2. Bestätigen Sie, dass Engineering-Benutzer nicht auf Marketing-Ressourcen zugreifen können:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n marketing-ns
```

Schritt 6: Zugriff auf AppVault-Objekte gewähren

Um Datenmanagementaufgaben wie Backups und Snapshots auszuführen, muss der Clusteradministrator einzelnen Benutzern Zugriff auf AppVault-Objekte gewähren.

Schritte

1. Erstellen und Anwenden einer AppVault- und geheimen YAML-Kombinationsdatei, die einem Benutzer Zugriff auf einen AppVault gewährt. Der folgende CR gewährt dem Benutzer beispielsweise Zugriff auf einen AppVault `eng-user`:

```

apiVersion: v1
data:
  accessKeyID: <ID_value>
  secretAccessKey: <key_value>
kind: Secret
metadata:
  name: appvault-for-eng-user-only-secret
  namespace: trident-protect
type: Opaque
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: appvault-for-eng-user-only
  namespace: trident-protect # Trident protect system namespace
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: testbucket
      endpoint: 192.168.0.1:30000
      secure: "false"
      skipCertValidation: "true"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: appvault-for-eng-user-only-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: appvault-for-eng-user-only-secret
  providerType: GenericS3

```

2. Erstellen und Anwenden eines Rollen-CR, damit Clusteradministratoren Zugriff auf bestimmte Ressourcen in einem Namespace gewähren können. Beispiel:

```

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: eng-user-appvault-reader
  namespace: trident-protect
rules:
- apiGroups:
  - protect.trident.netapp.io
  resourceNames:
  - appvault-for-enguser-only
  resources:
  - appvaults
  verbs:
  - get

```

3. Erstellen und wenden Sie einen RoleBinding CR an, um die Berechtigungen an den Benutzer eng-user zu binden. Beispiel:

```

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eng-user-read-appvault-binding
  namespace: trident-protect
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: eng-user-appvault-reader
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns

```

4. Überprüfen Sie, ob die Berechtigungen korrekt sind.

- a. Es wird versucht, die AppVault-Objektinformationen für alle Namespaces abzurufen:

```

kubectl get appvaults -n trident-protect
--as=system:serviceaccount:engineering-ns:eng-user

```

Sie sollten eine Ausgabe wie die folgende sehen:

```
Error from server (Forbidden): appvaults.protect.trident.netapp.io is
forbidden: User "system:serviceaccount:engineering-ns:eng-user"
cannot list resource "appvaults" in API group
"protect.trident.netapp.io" in the namespace "trident-protect"
```

- b. Testen Sie, ob der Benutzer die AppVault-Informationen erhalten kann, auf die er jetzt Zugriff hat:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get appvaults.protect.trident.netapp.io/appvault-for-eng-user-only -n
trident-protect
```

Sie sollten eine Ausgabe wie die folgende sehen:

```
yes
```

Ergebnis

Die Benutzer, denen Sie AppVault-Berechtigungen erteilt haben, sollten autorisierte AppVault-Objekte für Anwendungsdatenverwaltungsvorgänge verwenden können und nicht in der Lage sein, auf Ressourcen außerhalb der zugewiesenen Namespaces zuzugreifen oder neue Ressourcen zu erstellen, auf die sie keinen Zugriff haben.

Generieren Sie ein Trident Protect Supportpaket

Mit Trident Protect können Administratoren Bundles erstellen, die für die Unterstützung von NetApp nützliche Informationen enthalten, einschließlich Protokollen, Kennzahlen und Topologieinformationen zu den zu managenden Clustern und Apps. Wenn Sie mit dem Internet verbunden sind, können Sie Supportpakete mithilfe einer benutzerdefinierten Ressourcendatei (CR) auf die NetApp-Support-Website (NSS) hochladen.

Erstellen Sie mithilfe eines CR-Systems ein Supportpaket

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (z. B. `trident-protect-support-bundle.yaml`).
2. Konfigurieren Sie die folgenden Attribute:
 - **metadata.name:** (*required*) der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
 - **Spec.triggerType:** (*required*) legt fest, ob das Support-Bundle sofort generiert oder geplant wird. Die geplante Bundle-Generierung findet um 12:00 UHR UTC statt. Mögliche Werte:
 - Geplant
 - Manuell
 - **Spec.UploadEnabled:** (*Optional*) steuert, ob das Supportpaket nach der Generierung auf die NetApp-Support-Website hochgeladen werden soll. Wenn nicht angegeben, wird standardmäßig auf `false`. Mögliche Werte:
 - Richtig
 - False (Standard)
 - **Spec.dataWindowStart:** (*Optional*) Eine Datumstring im RFC 3339-Format, die das Datum und die Uhrzeit angibt, zu der das Fenster der im Support-Bundle enthaltenen Daten beginnen soll. Wenn nicht angegeben, ist die Standardeinstellung vor 24 Stunden. Das früheste Fensterdatum, das Sie angeben können, ist vor 7 Tagen.

Beispiel YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AutoSupportBundle
metadata:
  name: trident-protect-support-bundle
spec:
  triggerType: Manual
  uploadEnabled: true
  dataWindowStart: 2024-05-05T12:30:00Z
```

3. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `astra-support-bundle.yaml` haben, wenden Sie den CR an:

```
kubectl apply -f trident-protect-support-bundle.yaml
```

Erstellen Sie ein Support-Bundle mithilfe der CLI

Schritte

1. Erstellen Sie das Supportpaket, und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung. Der `trigger-type` legt fest, ob das Bündel sofort erstellt wird oder ob die

Erstellungszeit vom Zeitplan vorgegeben ist, und kann oder Scheduled sein Manual. Die Standardeinstellung ist Manual.

Beispiel:

```
tridentctl-protect create autosupportbundle <my_bundle_name>  
--trigger-type <trigger_type>
```

Upgrade von Trident Protect

Sie können ein Upgrade von Trident Protect auf die neueste Version durchführen, um von neuen Funktionen oder Fehlerkorrekturen zu profitieren.

Führen Sie zum Upgrade von Trident Protect die folgenden Schritte aus.

Schritte

1. Aktualisieren Sie das Trident Helm-Repository:

```
helm repo update
```

2. Aktualisieren Sie die Trident-Schutz-CRDs:

```
helm upgrade trident-protect-crds netapp-trident-protect/trident-  
protect-crds --version 100.2410.1 --namespace trident-protect
```

3. Upgrade-Trident-Schutz:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect  
--version 100.2410.1 --namespace trident-protect
```

Managen und sichern Sie Applikationen

Verwenden Sie zum Verwalten von Buckets Trident Protect AppVault-Objekte

Die benutzerdefinierte Bucket-Ressource (CR) für Trident Protect wird als AppVault bezeichnet. AppVault-Objekte sind die deklarative Kubernetes-Workflow-Darstellung eines Storage-Buckets. Ein AppVault CR enthält die Konfigurationen, die für einen Bucket erforderlich sind, der für Schutzvorgänge verwendet werden kann, z. B. Backups, Snapshots, Wiederherstellungsvorgänge und SnapMirror-Replikation. Nur Administratoren können AppVaults erstellen.

Beispiele für die Schlüsselgenerierung und AppVault-Definitionen

Beim Definieren eines AppVault CR müssen Sie Anmeldeinformationen eingeben, um auf die vom Anbieter gehosteten Ressourcen zugreifen zu können. Die Art und Weise, wie Sie die Schlüssel für die Anmeldeinformationen generieren, hängt vom Anbieter ab. Im Folgenden finden Sie Beispiele für die Schlüsselgenerierung über die Befehlszeile für mehrere Anbieter, gefolgt von AppVault-Beispieldefinitionen für jeden Anbieter.

Beispiele für die Schlüsselgenerierung

In den folgenden Beispielen können Sie Schlüssel für die Anmeldedaten der einzelnen Cloud-Provider erstellen.

Google Cloud

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

Amazon S3 (AWS)

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

Microsoft Azure

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

Allgemein S3

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

ONTAP S3

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

AppVault CR-Beispiele

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

Google Cloud

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  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-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

Microsoft Azure

```

apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret

```

Allgemein S3

```

apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: generic-s3-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3

```

ONTAP S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: OntapS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

StorageGRID S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket-b643cc50-0429-4ad5-
  971f-ac4a83621922
  namespace: trident-protect
spec:
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

Beispiele für die Erstellung von AppVault mithilfe der Trident Protect CLI

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

Google Cloud

```
tridentctl-protect create vault GCP my-new-vault --bucket mybucket  
--project my-gcp-project --secret <gcp-creds>/<credentials>
```

Amazon S3 (AWS)

```
tridentctl-protect create vault AWS <vault-name> --bucket <bucket-name>  
--secret <secret-name> --endpoint <s3-endpoint>
```

Microsoft Azure

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

Allgemein S3

```
tridentctl-protect create vault GenericS3 <vault-name> --bucket  
<bucket-name> --secret <secret-name> --endpoint <s3-endpoint>
```

ONTAP S3

```
tridentctl-protect create vault OntapS3 <vault-name> --bucket <bucket-  
name> --secret <secret-name> --endpoint <s3-endpoint>
```

StorageGRID S3

```
tridentctl-protect create vault StorageGridS3 s3vault --bucket <bucket-  
name> --secret <secret-name> --endpoint <s3-endpoint>
```

Verwenden Sie den AppVault-Browser, um Informationen zu AppVault anzuzeigen

Sie können das Trident Protect CLI-Plugin verwenden, um Informationen über AppVault-Objekte anzuzeigen, die auf dem Cluster erstellt wurden.

Schritte

1. Inhalt eines AppVault-Objekts anzeigen:

```
tridentctl-protect get appvaultcontent gcp-vault --show-resources all
```

Beispielausgabe:

```

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

```

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

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

Entfernen Sie einen AppVault

Sie können ein AppVault-Objekt jederzeit entfernen.



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

Bevor Sie beginnen

Stellen Sie sicher, dass Sie alle Snapshots und Backups gelöscht haben, die im zugehörigen Bucket gespeichert sind.

Entfernen Sie einen AppVault mithilfe der Kubernetes-CLI

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

```
kubectl delete appvault <appvault_name> -n trident-protect
```

Entfernen Sie einen AppVault mithilfe der Trident Protect-CLI

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

```
tridentctl-protect delete appvault <appvault_name> -n trident-protect
```

Definieren Sie mit Trident Protect eine Applikation für das Management

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

Erstellen Sie ein AppVault CR

Sie müssen einen AppVault CR erstellen, der bei der Durchführung von Datenschutzvorgängen auf der Anwendung verwendet wird. Der AppVault CR muss sich auf dem Cluster befinden, auf dem Trident Protect installiert ist. Der AppVault CR ist spezifisch für Ihre Umgebung. Beispiele für AppVault CRS finden Sie unter ["Benutzerdefinierte Ressourcen von AppVault."](#)

Definieren Sie eine Anwendung

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

Fügen Sie eine Anwendung mit einem CR hinzu

Schritte

1. Erstellen Sie die CR-Datei der Zielanwendung:
 - a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (z. B. `maria-app.yaml`).
 - b. Konfigurieren Sie die folgenden Attribute:
 - **metadata.name:** (*required*) der Name der benutzerdefinierten Ressource der Anwendung. Beachten Sie den von Ihnen ausgewählten Namen, da sich andere CR-Dateien, die für Schutzvorgänge benötigt werden, auf diesen Wert beziehen.
 - **spec.includedNamespaces:** (*required*) Verwenden Sie Namespace-Labels oder einen Namespace-Namen, um Namespaces anzugeben, in denen die Anwendungsressourcen vorhanden sind. Der Application Namespace muss Teil dieser Liste sein.

Beispiel YAML:

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

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

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

Fügen Sie eine Anwendung mithilfe der CLI hinzu

Schritte

1. Erstellen und wenden Sie die Anwendungsdefinition an, indem Sie Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Sie können Namespaces und Ressourcen in die Anwendungsdefinition mit kommasetrennten Listen mit den im folgenden Beispiel gezeigten Argumenten aufnehmen:

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

Sicherung von Applikationen mit Trident Protect

Sie können alle von Trident Protect gemanagten Applikationen schützen, indem Sie Snapshots und Backups mithilfe einer automatisierten Sicherungsrichtlinie oder ad hoc erstellen.



Sie können Trident Protect so konfigurieren, dass Dateisysteme während des Datenschutzes eingefroren und wieder eingefroren werden. ["Erfahren Sie mehr über das Konfigurieren des Dateisystemfrierens mit Trident Protect"](#).

Erstellen Sie einen On-Demand Snapshot

Sie können jederzeit einen On-Demand-Snapshot erstellen.

Erstellen Sie mithilfe eines CR-Systems einen Snapshot

Schritte

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

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Snapshot
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  reclaimPolicy: Delete
```

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

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

Erstellen Sie einen Snapshot mithilfe der CLI

Schritte

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

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

Erstellen Sie ein On-Demand-Backup

Sie können eine App jederzeit sichern.

Erstellen Sie ein Backup mit einem CR

Schritte

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

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

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

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

Erstellen Sie mithilfe der CLI ein Backup

Schritte

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

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

Erstellen Sie einen Zeitplan für die Datensicherung

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

Erstellen Sie einen Zeitplan mit einem CR

Schritte

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

```

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

```

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

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

Erstellen Sie einen Zeitplan über die CLI

Schritte

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



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

```

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

```

Löschen Sie einen Snapshot

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

Schritte

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

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

Löschen Sie ein Backup

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

Schritte

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

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

Überprüfen Sie den Status eines Sicherungsvorgangs

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

Schritte

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

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

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

Falls Sie Trident Protect installiert haben, können Sie die platzsparenden Backup- und Restore-Funktionen für Storage-Back-Ends aktivieren, die die Azure-NetApp-Files Storage-Klasse verwenden und vor Trident 24.06 erstellt wurden. Diese Funktion arbeitet mit NFSv4-Volumes zusammen und verbraucht keinen zusätzlichen Speicherplatz aus dem Kapazitäts-Pool.

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 begrenzte Schutzfunktionen, bis Sie diesen Vorgang abgeschlossen haben.
- Sie haben `azure-netapp-files` als Standard-Storage-Klasse für Ihr Storage-Back-End ausgewählt.

Erweitern Sie für Konfigurationsschritte

1. Gehen Sie in Trident folgendermaßen vor, wenn das ANF-Volume vor dem Upgrade auf Trident 24.10 erstellt wurde:
 - a. Aktivieren Sie das Snapshot-Verzeichnis für jedes PV, das auf Azure-NetApp-Dateien basiert und der Anwendung zugeordnet ist:

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

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

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

Antwort:

```
snapshotDirectory: "true"
```

+

Wenn das Snapshot-Verzeichnis nicht aktiviert ist, wählt Trident Protect die regelmäßige Backup-Funktion aus, die während des Backup-Prozesses vorübergehend Speicherplatz im Kapazitäts-Pool verbraucht. Stellen Sie in diesem Fall sicher, dass im Kapazitätspool ausreichend Speicherplatz verfügbar ist, um ein temporäres Volume der Größe des zu sichernden Volumes zu erstellen.

Ergebnis

Die Applikation ist mit Trident Protect für die Sicherung und Wiederherstellung bereit. Jede PVC kann auch von anderen Anwendungen für Backups und Wiederherstellungen verwendet werden.

Stellen Sie Applikationen mit Trident Protect wieder her

Sie können Trident Protect verwenden, um Ihre Anwendung aus einem Snapshot oder einem Backup wiederherzustellen. Das Wiederherstellen aus einem vorhandenen Snapshot erfolgt schneller, wenn die Anwendung auf dasselbe Cluster wiederhergestellt wird.



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

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

Während von Restore- und Failover-Vorgängen werden Labels und Annotationen im Ziel-Namespace so angefertigt, dass sie den Labels und Annotationen im Quell-Namespace entsprechen. Beschriftungen oder

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



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

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

```
kubectl set env -n trident-protect deploy/trident-protect-controller-manager
RESTORE_SKIP_NAMESPACE_ANNOTATIONS=<annotation_key_to_skip_1>,<annotation_key_to_skip_2>
```

Wenn Sie die Quellanwendung mit Helm mit dem Flag installiert haben, wird der Label-Schlüssel mit `--create-namespace` einer speziellen Bestrahlung `name` versehen. Während des Wiederherstellungs- oder Failover-Prozesses kopiert Trident Protect dieses Label in den Ziel-Namespace, aktualisiert jedoch den Wert auf den Ziel-Namespace-Wert, wenn der Wert von Quell-Namespace mit dem Quell-Namespace übereinstimmt. Wenn dieser Wert nicht mit dem Quell-Namespace übereinstimmt, wird er ohne Änderungen in den Ziel-Namespace kopiert.

Beispiel

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

Vor der Wiederherstellung oder dem Failover

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

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

Nach dem Wiederherstellungsvorgang

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

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

Wiederherstellung aus einem Backup in einem anderen Namespace

Wenn Sie ein Backup mit einem Backup Restore 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. Erstellen Sie zum Schutz der wiederhergestellten Applikation On-Demand-Backups oder Snapshots oder erstellen Sie einen Sicherungszeitplan.



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

CR verwenden

Schritte

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

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

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

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

3. (*Optional*) Wenn Sie nur bestimmte Ressourcen der wiederherzustellenden Anwendung auswählen müssen, fügen Sie eine Filterung hinzu, die Ressourcen mit bestimmten Bezeichnungen enthält oder ausschließt:
 - **ResourceFilter.resourceSelectionCriteria:** (Erforderlich für die Filterung) Verwenden `Include` oder `Exclude` um eine in `resourceMatchers` definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden `resourceMatchers`-Parameter hinzu, um die einzuschließenden oder auszuschließenden Ressourcen zu definieren:

- **RefindeFilter.refindeMatchers:** Eine Reihe von refineMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, stimmen sie als OP-Operation überein, und die Felder innerhalb jedes Elements (Gruppe, Typ, Version) stimmen mit einer UND-Operation überein.
 - **ResourceMatchers[].Group:** (*Optional*) Gruppe der zu filternden Ressource.
 - **ResourceMatchers[].Kind:** (*Optional*) Art der zu filternden Ressource.
 - **ResourceMatchers[].Version:** (*Optional*) Version der zu filternden Ressource.
 - **ResourceMatchers[].Namen:** (*Optional*) Namen im Kubernetes metadata.name-Feld der zu filternden Ressource.
 - **ResourceMatchers[].Namespaces:** (*Optional*) Namespaces im Kubernetes metadata.name-Feld der zu filternden Ressource.
 - **ResourceMatchers[].labelSelectors:** (*Optional*) Label selector string im Feld Kubernetes metadata.name der Ressource, wie im definiert "[Kubernetes-Dokumentation](#)". Zum Beispiel: "trident.netapp.io/os=linux".

Beispiel:

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

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

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

Verwenden Sie die CLI

Schritte

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

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

Wiederherstellung von einem Backup in den ursprünglichen Namespace

Sie können ein Backup im ursprünglichen Namespace jederzeit wiederherstellen.

CR verwenden

Schritte

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

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

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

Beispiel:

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

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

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

Beispiel:

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

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

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

Verwenden Sie die CLI

Schritte

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

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

Wiederherstellung von einem Backup in einem anderen Cluster

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

Bevor Sie beginnen

Stellen Sie sicher, dass die folgenden Voraussetzungen erfüllt sind:

- Auf dem Zielcluster ist Trident Protect installiert.
- Der Zielcluster hat Zugriff auf den Bucket-Pfad desselben AppVault wie das Quellcluster, in dem das Backup gespeichert ist.

Schritte

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

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



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

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

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

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

Beispielausgabe:

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

```

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

CR verwenden

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

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



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

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

Beispiel:

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

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

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

Verwenden Sie die CLI

1. Verwenden Sie den folgenden Befehl, um die Anwendung wiederherzustellen und Werte in Klammern durch Informationen aus Ihrer Umgebung zu ersetzen. Das Namespace-Mapping-Argument verwendet durch Doppelpunkte getrennte Namespaces, um Quellnamespaces im Format

source1:dest1,source2:dest2 den korrekten Zielnamespace zuzuordnen. Beispiel:

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

Wiederherstellung von einem Snapshot in einem anderen Namespace

Sie können Daten aus einem Snapshot mithilfe einer benutzerdefinierten Ressourcendatei (CR) entweder in einem anderen Namespace oder im ursprünglichen QuellNamespace wiederherstellen. Wenn Sie einen Snapshot mithilfe eines SnapshotRestore CR in einem anderen Namespace wiederherstellen, stellt Trident Protect die Anwendung in einem neuen Namespace wieder her und erstellt eine Anwendungs-CR für die wiederhergestellte Anwendung. Erstellen Sie zum Schutz der wiederhergestellten Applikation On-Demand-Backups oder Snapshots oder erstellen Sie einen Sicherungszeitplan.

CR verwenden

Schritte

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

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

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

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

3. (*Optional*) Wenn Sie nur bestimmte Ressourcen der wiederherzustellenden Anwendung auswählen müssen, fügen Sie eine Filterung hinzu, die Ressourcen mit bestimmten Bezeichnungen enthält oder ausschließt:
 - **ResourceFilter.resourceSelectionCriteria:** (Erforderlich für die Filterung) Verwenden `Include` oder `Exclude` um eine in `resourceMatchers` definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden `resourceMatchers`-Parameter hinzu, um die einzuschließenden oder

auszuschließenden Ressourcen zu definieren:

- **RefindeFilter.refindeMatchers:** Eine Reihe von refineMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, stimmen sie als OP-Operation überein, und die Felder innerhalb jedes Elements (Gruppe, Typ, Version) stimmen mit einer UND-Operation überein.
 - **ResourceMatchers[].Group:** (*Optional*) Gruppe der zu filternden Ressource.
 - **ResourceMatchers[].Kind:** (*Optional*) Art der zu filternden Ressource.
 - **ResourceMatchers[].Version:** (*Optional*) Version der zu filternden Ressource.
 - **ResourceMatchers[].Namen:** (*Optional*) Namen im Kubernetes metadata.name-Feld der zu filternden Ressource.
 - **ResourceMatchers[].Namespaces:** (*Optional*) Namespaces im Kubernetes metadata.name-Feld der zu filternden Ressource.
 - **ResourceMatchers[].labelSelectors:** (*Optional*) Label selector string im Feld Kubernetes metadata.name der Ressource, wie im definiert ["Kubernetes-Dokumentation"](#). Zum Beispiel: "trident.netapp.io/os=linux".

Beispiel:

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

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

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

Verwenden Sie die CLI

Schritte

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

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

Beispiel:

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

Wiederherstellung von einem Snapshot im ursprünglichen Namespace

Sie können einen Snapshot jederzeit im ursprünglichen Namespace wiederherstellen.

CR verwenden

Schritte

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

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

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

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

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

Beispiel:

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

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

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

Verwenden Sie die CLI

Schritte

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

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

Überprüfen Sie den Status eines Wiederherstellungsvorgangs

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

Schritte

1. Verwenden Sie den folgenden Befehl, um den Status des Wiederherstellungsvorgangs abzurufen und

Werte in Bracken durch Informationen aus Ihrer Umgebung zu ersetzen:

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

Replizieren Sie Applikationen mit NetApp SnapMirror und Trident Protect

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

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

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



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

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

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

Wenn Sie die Quellanwendung mit Helm mit dem Flag installiert haben, wird der Label-Schlüssel mit `--create-namespace` einer speziellen Bestrahlung `name` versehen. Während des Wiederherstellungs- oder Failover-Prozesses kopiert Trident Protect dieses Label in den Ziel-Namespace, aktualisiert jedoch den Wert auf den Ziel-Namespace-Wert, wenn der Wert von Quell-Namespace mit dem Quell-Namespace übereinstimmt. Wenn dieser Wert nicht mit dem Quell-Namespace übereinstimmt, wird er ohne Änderungen in den Ziel-Namespace kopiert.

Beispiel

Im folgenden Beispiel wird ein Quell- und Ziel-Namespace präsentiert, der jeweils unterschiedliche Annotationen und Labels enthält. Sie können den Status des Ziel-Namespace vor und nach dem Vorgang

anzeigen und sehen, wie die Annotationen und Labels im Ziel-Namespace kombiniert oder überschrieben werden.

Vor der Wiederherstellung oder dem Failover

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

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

Nach dem Wiederherstellungsvorgang

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

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



Sie können Trident Protect so konfigurieren, dass Dateisysteme während des Datenschutzes eingefroren und wieder eingefroren werden. ["Erfahren Sie mehr über das Konfigurieren des Dateisystemfrierens mit Trident Protect"](#).

Richten Sie eine Replikationsbeziehung ein

Die Einrichtung einer Replikationsbeziehung umfasst Folgendes:

- Festlegen der Häufigkeit, die Trident schützen soll, um einen App-Snapshot zu erstellen (was die Kubernetes-Ressourcen der Applikation sowie die Volume-Snapshots für die Volumes der Applikation umfasst)
- Auswahl des Replizierungszeitplans (einschließlich Kubernetes-Ressourcen und persistenter Volume-Daten)
- Einstellen der Uhrzeit für die Erstellung des Snapshots

Schritte

1. Erstellen Sie einen AppVault für die Quellanwendung auf dem Quellcluster. Ändern Sie abhängig von Ihrem Storage-Anbieter ein Beispiel in ["Benutzerdefinierte Ressourcen von AppVault"](#) entsprechend Ihrer

Umgebung:

Erstellen Sie einen AppVault mit einem CR

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

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

Erstellen Sie einen AppVault mithilfe der CLI

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

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

2. Erstellen Sie die CR-Quellanwendung:

Erstellen Sie die Quellanwendung mit einem CR

- a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (z. B. `trident-protect-app-source.yaml`).
- b. Konfigurieren Sie die folgenden Attribute:
 - **metadata.name:** (*required*) der Name der benutzerdefinierten Ressource der Anwendung. Notieren Sie sich den ausgewählten Namen, da sich andere CR-Dateien, die für eine Replikationsbeziehung benötigt werden, auf diesen Wert beziehen.
 - **spec.includedNamespaces:** (*required*) ein Array von Namespaces und zugehörigen Labels. Verwenden Sie Namespace-Namen und Grenzen Sie optional den Umfang der Namespaces mit Beschriftungen ein, um Ressourcen anzugeben, die in den hier aufgeführten Namespaces vorhanden sind. Der Application Namespace muss Teil dieses Arrays sein.

Beispiel YAML:

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

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

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

Erstellen Sie die Quellanwendung mithilfe der CLI

- a. Erstellen Sie die Quellanwendung. Beispiel:

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

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

Erstellen Sie einen Schnappschuss mit einem CR-System

a. Erstellen Sie einen Replikationszeitplan für die Quellanwendung:

- i. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (z. B. `trident-protect-schedule.yaml`).
- ii. Konfigurieren Sie die folgenden Attribute:
 - **metadata.name:** (*required*) der Name der benutzerdefinierten Ressource für den Zeitplan.
 - **Spec.AppVaultRef:** (*required*) dieser Wert muss mit dem Feld `metadata.name` des AppVault für die Quellanwendung übereinstimmen.
 - **Spec.ApplicationRef:** (*required*) dieser Wert muss mit dem Feld `metadata.name` der Quellanwendung CR übereinstimmen.
 - **Spec.backupRetention:** (*required*) Dieses Feld ist erforderlich und der Wert muss auf 0 gesetzt werden.
 - **Spec.enabled:** Muss auf `true` gesetzt werden.
 - **spec.granularity:** muss auf eingestellt sein `Custom`.
 - **Spec.recurrenceRule:** Definieren Sie ein Startdatum in UTC-Zeit und ein Wiederholungsintervall.
 - **Spec.snapshotRetention:** Muss auf 2 gesetzt werden.

Beispiel YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule-0elf88ab-f013-4bce-8ae9-6afed9df59a1
  namespace: my-app-namespaces
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-46a3-420a-b351-45795e1b5e34
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

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

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

Erstellen Sie einen Snapshot mit der CLI

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

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

4. Erstellen Sie eine Quellenanwendung AppVault CR auf dem Zielcluster, die mit dem AppVault CR identisch ist, den Sie auf das Quellcluster angewendet haben, und benennen Sie es (z. B. trident-protect-appvault-primary-destination.yaml).

5. CR anwenden:

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

6. Erstellen Sie einen AppVault für die Zielanwendung auf dem Zielcluster. Ändern Sie abhängig von Ihrem Storage-Anbieter ein Beispiel in ["Benutzerdefinierte Ressourcen von AppVault"](#) entsprechend Ihrer Umgebung:

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

- b. Konfigurieren Sie die folgenden Attribute:

- **metadata.name:** (*required*) der Name der benutzerdefinierten AppVault-Ressource. Notieren Sie sich den ausgewählten Namen, da sich andere CR-Dateien, die für eine Replikationsbeziehung benötigt werden, auf diesen Wert beziehen.
- **spec.providerConfig:** (*required*) speichert die Konfiguration, die für den Zugriff auf den AppVault unter Verwendung des angegebenen Providers erforderlich ist. Wählen Sie eine `bucketName` und alle anderen notwendigen Details für Ihren Provider. Notieren Sie sich die ausgewählten Werte, da sich andere CR-Dateien, die für eine Replikationsbeziehung benötigt werden, auf diese Werte beziehen. Beispiele für AppVault CRS mit anderen Anbietern finden Sie unter ["Benutzerdefinierte Ressourcen von AppVault"](#).
- **spec.providerCredentials:** (*required*) speichert Verweise auf alle Anmeldeinformationen, die für den Zugriff auf AppVault unter Verwendung des angegebenen Anbieters erforderlich sind.
 - **spec.providerCredentials.valueFromSecret:** (*required*) gibt an, dass der Wert der Zugangsdaten von einem Secret stammen soll.
 - **Key:** (*required*) der gültige Schlüssel des zu wählenden Geheimnisses.
 - **Name:** (*required*) Name des Geheimnisses, das den Wert für dieses Feld enthält. Muss sich im gleichen Namespace befinden.

- **spec.providerCredentials.secretAccessKey:** (*required*) der Zugriffsschlüssel, der für den Zugriff auf den Provider verwendet wird. Der **Name** sollte mit **spec.providerCredentials.valueFromSecret.name** übereinstimmen.
 - **spec.providerType:** (*required*) legt fest, was das Backup bereitstellt, z. B. NetApp ONTAP S3, generisches S3, Google Cloud oder Microsoft Azure. Mögliche Werte:
 - AWS
 - Azure
 - GCP
 - Generisch-s3
 - ONTAP-s3
 - StorageGRID-s3
- c. Nachdem Sie die Datei mit den richtigen Werten ausgefüllt `trident-protect-appvault-secondary-destination.yaml` haben, wenden Sie den CR an:

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

7. Erstellen Sie eine AppMirrorRelationship CR-Datei:

Erstellen Sie eine AppMirrorRelationship mithilfe eines CR

- a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (z. B. `trident-protect-relationship.yaml`).
- b. Konfigurieren Sie die folgenden Attribute:
 - **metadata.name:** (erforderlich) der Name der benutzerdefinierten AppMirrorRelationship-Ressource.
 - **spec.destinationAppVaultRef:** (*required*) dieser Wert muss mit dem Namen des AppVault für die Zielanwendung auf dem Zielcluster übereinstimmen.
 - **spec.namespaceMapping:** (*required*) der Ziel- und Quellnamespaces muss mit dem im jeweiligen Anwendungs-CR definierten AnwendungsNamespace übereinstimmen.
 - **Spec.sourceAppVaultRef:** (*required*) dieser Wert muss mit dem Namen des AppVault für die Quellanwendung übereinstimmen.
 - **Spec.sourceApplicationName:** (*required*) dieser Wert muss mit dem Namen der Quellanwendung übereinstimmen, die Sie in der Quellanwendung CR definiert haben.
 - **Spec.storageClassName:** (*required*) Wählen Sie den Namen einer gültigen Storage-Klasse auf dem Cluster. Die Storage-Klasse muss mit einer ONTAP Storage-VM verknüpft werden, die mit der Quellumgebung peert wird.
 - **Spec.recurrenceRule:** Definieren Sie ein Startdatum in UTC-Zeit und ein Wiederholungsintervall.

Beispiel YAML:

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

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

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

Erstellen Sie eine AppMirrorRelationship mithilfe der CLI

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

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

8. (Optional) Status und Status der Replikationsbeziehung prüfen:

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

Failover zum Ziel-Cluster

Mit Trident Protect können Sie ein Failover replizierter Applikationen auf ein Ziel-Cluster durchführen. Durch dieses Verfahren wird die Replikationsbeziehung angehalten und die App wird auf dem Ziel-Cluster online geschaltet. Trident Protect stoppt die App auf dem Quell-Cluster nicht, wenn sie betriebsbereit war.

Schritte

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

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

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

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

Neusynchronisierung einer fehlgeschlagenen Replikationsbeziehung

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

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



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

Schritte

1. Erstellen Sie einen Snapshot der Quellanwendung.
2. Öffnen Sie die AppMirrorRelationship CR-Datei (z. B. `trident-protect-relationship.yaml`) und ändern Sie den Wert von `spec.desiredState` in `Established`.
3. Speichern Sie die CR-Datei.
4. CR anwenden:

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

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

Reverse Resynchronisierung einer Failover-Replizierungsbeziehung

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

Schritte

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

Richtung der Anwendungsreplikation umkehren

Wenn Sie die Replizierungsrichtung umkehren, verschiebt Trident Protect die Applikation auf das Ziel-Storage-Back-End, während die Replikation zurück zum ursprünglichen Quell-Storage-Back-End fortgesetzt wird. Trident Protect stoppt die Quellapplikation und repliziert die Daten zum Ziel, bevor ein Failover zur Zielapplikation erfolgt.

In dieser Situation tauschen Sie Quelle und Ziel aus.

Schritte

1. Erstellen Sie einen Snapshot zum Herunterfahren:

Erstellen Sie einen Snapshot zum Herunterfahren mit einem CR

- a. Deaktivieren Sie die Schutzrichtlinienpläne für die Quellenanwendung.
- b. Erstellen Sie eine CR-Datei für den ShutdownSnapshot:
 - i. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (z. B. `trident-protect-shutdownsnapshot.yaml`).
 - ii. Konfigurieren Sie die folgenden Attribute:
 - **metadata.name:** (*required*) der Name der benutzerdefinierten Ressource.
 - **Spec.AppVaultRef:** (*required*) dieser Wert muss mit dem Feld `metadata.name` des AppVault für die Quellenanwendung übereinstimmen.
 - **Spec.ApplicationRef:** (*required*) dieser Wert muss mit dem Feld `metadata.name` der CR-Datei der Quellenanwendung übereinstimmen.

Beispiel YAML:

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

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

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

Erstellen Sie einen Snapshot zum Herunterfahren über die CLI

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

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

2. Nachdem der Snapshot abgeschlossen ist, rufen Sie den Status des Snapshots ab:

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

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

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

4. Führen Sie mit der folgenden Änderung ein Failover vom Ziel-Cluster zum Quell-Cluster durch:



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

5. Führen Sie die Schritte für die umgekehrte Resynchronisierung in [Reverse Resynchronisierung einer Failover-Replizierungsbeziehung](#) aus.

6. Aktivieren Sie Schutzzeitpläne auf dem neuen Quellcluster.

Ergebnis

Die folgenden Aktionen werden aufgrund der umgekehrten Replikation durchgeführt:

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

Führen Sie ein Failback von Anwendungen auf das ursprüngliche Quellcluster durch

Mithilfe von Trident Protect können Sie nach einem Failover-Vorgang ein Failback durchführen, indem Sie die folgende Sequenz von Vorgängen verwenden. In diesem Workflow zur Wiederherstellung der ursprünglichen Replikationsrichtung repliziert Trident Protect alle Anwendungsänderungen zurück zur ursprünglichen Quellanwendung, bevor die Replikationsrichtung rückgängig gemacht wird.

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

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



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

- Kehren Sie die Replikationsrichtung um.

Schritte

1. Führen Sie die [Reverse Resynchronisierung einer Failover-Replizierungsbeziehung](#) Schritte aus.
2. Führen Sie die [Richtung der Anwendungsreplikation umkehren](#) Schritte aus.

Löschen einer Replikationsbeziehung

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

Schritte

1. Löschen Sie die AppMirrorRelationship-CR:

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

Migrieren Sie Applikationen mit Trident Protect

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



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

Backup- und Restore-Vorgänge

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

Klonen auf dasselbe Cluster

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

Schritte

1. Führen Sie einen der folgenden Schritte aus:
 - a. ["Erstellen Sie einen Snapshot"](#).
 - b. ["Erstellen Sie ein Backup"](#).

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

Klonen auf anderes Cluster

Um eine Anwendung auf einem anderen Cluster zu klonen (einen Cluster-übergreifenden Klon durchführen), erstellen Sie ein Backup auf dem Quellcluster, und stellen Sie das Backup dann auf einem anderen Cluster wieder her. Stellen Sie sicher, dass Trident Protect auf dem Ziel-Cluster installiert ist.



Sie können eine Anwendung zwischen verschiedenen Clustern mit replizieren "[SnapMirror Replizierung](#)".

Schritte

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

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

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

Beispiel: (Ohne die Geheimnisse aus dem Wiederherstellungs-CR):

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

Stellen Sie den Snapshot mithilfe eines CR-Systems wieder her

Schritte

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

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

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

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

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

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

Beispiel:

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

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

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

Stellen Sie den Snapshot mithilfe der CLI wieder her

Schritte

1. Stellen Sie den Snapshot in einem anderen Namespace wieder her und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer Umgebung.
 - Das snapshot Argument verwendet einen Namespace und Snapshot-Namen im Format <namespace>/<name>.
 - Das namespace-mapping Argument verwendet durch Doppelpunkte getrennte Namespaces, um Quellnamespaces im Format den richtigen Zielnamespaces zuzuordnen source1:dest1, source2:dest2.

Beispiel:

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

Managen von Trident Protect-Ausführungshaken

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

Arten von Ausführungshaken

Trident Protect unterstützt die folgenden Typen von Ausführungshaken, je nachdem, wann sie ausgeführt werden können:

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

Ausführungsreihenfolge

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

1. Alle entsprechenden benutzerdefinierten Testhaken für die Ausführung vor dem Betrieb werden auf den entsprechenden Containern ausgeführt. Sie können beliebig viele benutzerdefinierte Hooks für die Vorbedienung erstellen und ausführen, aber die Reihenfolge der Ausführung dieser Haken vor der Operation ist weder garantiert noch konfigurierbar.
2. Das Dateisystem hängt sich ggf. ein. ["Erfahren Sie mehr über das Konfigurieren des Dateisystemfrierens mit Trident Protect"](#).
3. Der Vorgang der Datensicherung wird durchgeführt.
4. Eingefrorene Dateisysteme werden gegebenenfalls nicht eingefroren.
5. Alle entsprechenden benutzerdefinierten Testhaken für die Ausführung nach der Operation werden auf den entsprechenden Containern ausgeführt. Sie können beliebig viele benutzerdefinierte Haken für die Nachbearbeitung erstellen und ausführen, aber die Reihenfolge der Ausführung dieser Haken nach der Operation ist weder garantiert noch konfigurierbar.

Wenn Sie mehrere Testausführungshaken desselben Typs erstellen (z. B. Pre-Snapshot), ist die Reihenfolge der Ausführung dieser Haken nicht garantiert. Die Reihenfolge der Ausführung von Haken unterschiedlicher Art ist jedoch garantiert. Im Folgenden wird beispielsweise die Reihenfolge der Ausführung einer Konfiguration

beschrieben, die alle verschiedenen Hooks umfasst:

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



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



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

Wichtige Hinweise zu benutzerdefinierten Testausführungshaken

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

- Ein Testsuite muss ein Skript verwenden, um Aktionen durchzuführen. Viele Testsuitehooks können auf dasselbe Skript verweisen.
- Trident Protect erfordert, dass die Skripte, die Ausführungshaken verwenden, im Format ausführbarer Shell-Skripte geschrieben werden.
- Die Skriptgröße ist auf 96 KB begrenzt.
- Trident Protect verwendet Execution Hook-Einstellungen und alle übereinstimmenden Kriterien, um zu ermitteln, welche Hooks für einen Snapshot-, Backup- oder Wiederherstellungsvorgang gelten.



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

Filter für Testausführungshaken

Wenn Sie einen Ausführungshaken für eine Anwendung hinzufügen oder bearbeiten, können Sie dem Ausführungshaken Filter hinzufügen, um zu verwalten, welche Container der Hook entsprechen soll. Filter sind für Applikationen nützlich, die in allen Containern dasselbe Container-Image nutzen. Jedes Image kann jedoch für einen anderen Zweck (wie Elasticsearch) verwendet werden. Mit Filtern können Sie Szenarien erstellen, in denen Ausführungshaken auf einigen, aber nicht unbedingt allen identischen Containern ausgeführt werden. Wenn Sie mehrere Filter für einen einzelnen Testausführungshaken erstellen, werden diese mit einem logischen UND einem Operator kombiniert. Pro Testsuite können Sie bis zu 10 aktive Filter haben.

Jeder Filter, den Sie einem Execution Hook hinzufügen, verwendet einen regulären Ausdruck, um Container in Ihrem Cluster zu entsprechen. Wenn ein Haken einem Container entspricht, führt der Haken sein zugehöriges Skript auf diesem Container aus. Reguläre Ausdrücke für Filter verwenden die Syntax des regulären

Ausdrucks 2 (RE2), die das Erstellen eines Filters nicht unterstützt, der Container aus der Liste der Übereinstimmungen ausschließt. Informationen zur Syntax, die Trident Protect für reguläre Ausdrücke in Ausführungshook-Filtern unterstützt, finden Sie unter "[Syntaxunterstützung für regulären Ausdruck 2 \(RE2\)](#)".



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

Beispiele für Testausführungshaken

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

Erstellen Sie einen Ausführungshaken

Mit Trident Protect können Sie einen benutzerdefinierten Ausführungshaken für eine App erstellen. Sie müssen über die Berechtigungen Eigentümer, Administrator oder Mitglied verfügen, um Testausführungshaken zu erstellen.

CR verwenden

Schritte

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

Beispiel YAML:

```
apiVersion: protect.trident.netapp.io/v1
kind: ExecHook
metadata:
  name: example-hook-cr
  namespace: my-app-namespace
  annotations:
    astra.netapp.io/astra-control-hook-source-id:
/account/test/hookSource/id
spec:
  applicationRef: my-app-name
  stage: Pre
  action: Snapshot
  enabled: true
  hookSource: IyEvYmluL2Jhc2gKZWNoYAiZXhhbXBsZSBzY3JpcHQiCg==
  timeout: 10
  arguments:
    - FirstExampleArg
    - SecondExampleArg
  matchingCriteria:
    - type: containerName
      value: mysql
    - type: containerImage
      value: bitnami/mysql
    - type: podName
      value: mysql
    - type: namespaceName
      value: mysql-a
    - type: podLabel
      value: app.kubernetes.io/component=primary
    - type: podLabel
      value: helm.sh/chart=mysql-10.1.0
    - type: podLabel
      value: deployment-type=production
```

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

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

Verwenden Sie die CLI

Schritte

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

```
tridentctl-protect create 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>
```

Deinstallieren Sie Trident Protect

Möglicherweise müssen Sie Trident Protect-Komponenten entfernen, wenn Sie ein Upgrade von einer Testversion auf eine Vollversion des Produkts durchführen.

So entfernen Sie Trident Protect:

Schritte

1. Entfernen Sie die Trident Protect CR-Dateien:

```
helm uninstall -n trident-protect trident-protect-crds
```

2. Trident Protect entfernen:

```
helm uninstall -n trident-protect trident-protect
```

3. Entfernen Sie den Trident Protect Namespace:

```
kubectl delete ns trident-protect
```

Copyright-Informationen

Copyright © 2025 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.