Schützen Sie Anwendungen mit Trident Protect

Trident

NetApp
July 01, 2026

Inhalt

Schützen Sie Anwendungen mit Trident Protect . . . 1
- Erfahren Sie mehr über Trident Protect . . . 1
  - Was kommt als Nächstes? . . . 1
- Installieren Sie Trident Protect . . . 1
  - Trident Protect Anforderungen . . . 1
  - Trident Protect installieren und konfigurieren . . . 5
  - Installieren Sie das Trident Protect CLI-Plugin . . . 9
  - Trident Protect-Installation anpassen . . . 13
- Trident Protect verwalten . . . 18
  - Trident Protect-Autorisierung und Zugriffskontrolle verwalten . . . 18
  - Überwachen Sie die Trident Protect-Ressourcen . . . 25
  - Generieren Sie ein Trident Protect Support-Bundle . . . 30
  - Trident Protect aktualisieren . . . 32
- Anwendungen verwalten und schützen . . . 34
  - Verwenden Sie Trident Protect AppVault-Objekte, um Buckets zu verwalten . . . 34
  - Definieren Sie eine Anwendung für das Management mit Trident Protect . . . 48
  - Schützen Sie Anwendungen mit Trident Protect . . . 53
  - Anwendungen wiederherstellen . . . 66
  - Anwendungen mit NetApp SnapMirror und Trident Protect replizieren . . . 84
  - Anwendungen mit Trident Protect migrieren . . . 101
  - Trident Protect-Ausführungshooks verwalten . . . 105
- Trident Protect deinstallieren . . . 117

Schützen Sie Anwendungen mit Trident Protect

Erfahren Sie mehr über Trident Protect

NetApp Trident Protect bietet erweiterte Funktionen für das Management von Anwendungsdaten, die die Funktionalität und Verfügbarkeit zustandsbehafteter Kubernetes-Anwendungen verbessern, die durch NetApp ONTAP-Speichersysteme und den NetApp Trident CSI Storage Provisioner unterstützt werden. Trident Protect vereinfacht die Verwaltung, den Schutz und die Migration containerisierter Workloads zwischen Public Clouds und On-Premises-Umgebungen. Es bietet außerdem 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?

Sie können sich über die Anforderungen von Trident Protect informieren, bevor Sie es installieren:

- "Trident Protect Anforderungen"

Installieren Sie Trident Protect

Trident Protect Anforderungen

Beginnen Sie mit der Überprüfung der Einsatzbereitschaft Ihrer Betriebsumgebung, Anwendungscluster, Anwendungen und Lizenzen. Stellen Sie sicher, dass Ihre Umgebung diese Anforderungen erfüllt, um Trident Protect bereitzustellen und zu betreiben.

Trident Protect Kubernetes-Cluster-Kompatibilität

Trident Protect ist mit einer Vielzahl von vollständig verwalteten und selbstverwalteten Kubernetes-Angeboten kompatibel, einschließlich:

- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- Microsoft Azure Kubernetes Service (AKS)
- Red Hat OpenShift
- SUSE Harvester 1.7.0 (ONTAP iSCSI)
- SUSE Rancher
- VMware Tanzu Portfolio
- Upstream Kubernetes

- Trident Protect-Backups werden nur auf Linux-Rechenknoten unterstützt. Windows-Rechenknoten werden für Backup-Vorgänge nicht unterstützt.
- Stellen Sie sicher, dass der Cluster, auf dem Sie Trident Protect installieren, mit einem laufenden Snapshot-Controller und den zugehörigen CRDs konfiguriert ist. Informationen zur Installation eines Snapshot-Controllers finden Sie unter "diese Anweisungen".
- Stellen Sie sicher, dass mindestens eine VolumeSnapshotClass existiert. Weitere Informationen finden sich unter "VolumeSnapshotClass".
- Für die Installation von Trident Protect wird Helm 4.x oder höher benötigt.

Trident Protect Speicher-Backend-Kompatibilität

Trident Protect unterstützt die folgenden Storage-Backends:

- Amazon FSx for NetApp ONTAP
- Cloud Volumes ONTAP
- ONTAP Speicherarrays
- Google Cloud NetApp Volumes
- Azure NetApp Files

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

- Stellen Sie sicher, dass der mit dem Cluster verbundene NetApp Speicher Trident 24.02 oder neuer verwendet (Trident 24.10 wird empfohlen).
- Stellen Sie sicher, dass Sie über ein NetApp ONTAP storage backend verfügen.
- Stellen Sie sicher, dass Sie einen Objektspeicher-Bucket für das Speichern von Backups konfiguriert haben.
- Erstellen Sie alle Anwendungs-Namespaces, die Sie für Anwendungen oder Anwendungsdatenverwaltungsoperationen verwenden möchten. Trident Protect erstellt diese Namespaces nicht für Sie; wenn Sie in einer benutzerdefinierten Ressource einen nicht vorhandenen Namespace angeben, schlägt der Vorgang fehl.

Anforderungen für nas-economy Volumes

Trident Protect unterstützt Backup- und Wiederherstellungsvorgänge auf nas-economy Volumes. Snapshots, Klone und SnapMirror Replikation auf nas-economy Volumes werden derzeit nicht unterstützt. Sie müssen für jedes nas-economy Volume, das Sie mit Trident Protect verwenden möchten, ein Snapshot-Verzeichnis aktivieren.

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 Storage-System ausführen:

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

Sie können das Snapshot-Verzeichnis aktivieren, indem Sie für jedes nas-economy-Volume den folgenden Befehl ausführen und dabei `<volume-UUID>` durch die UUID des Volumes ersetzen, 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 `snapshotDir` auf `true` setzen. Vorhandene Volumes sind davon nicht betroffen.

Schutz von Daten mit KubeVirt VMs

Trident Protect bietet Funktionen zum Einfrieren und Auftauen des Dateisystems für KubeVirt virtuelle Maschinen während Datensicherungsoperationen, um die Datenkonsistenz sicherzustellen. Die Konfigurationsmethode und das Standardverhalten für VM-Einfrieroperationen variieren je nach Trident Protect-Version, wobei neuere Versionen eine vereinfachte Konfiguration über Helm-Chart-Parameter bieten.

Während Wiederherstellungsvorgängen werden alle `VirtualMachineSnapshots` für eine virtuelle Maschine (VM) erstellten Dateien nicht wiederhergestellt.

Trident Protect 25.10 und neuer

Trident Protect friert KubeVirt-Dateisysteme während der Datensicherungsoperationen automatisch ein und taut sie wieder auf, um Konsistenz zu gewährleisten. Ab Trident Protect 25.10 können Sie dieses Verhalten mit dem `vm.freeze` Parameter während der Helm-Chart-Installation deaktivieren. Der Parameter ist standardmäßig aktiviert.

```
helm install ... --set vm.freeze=false ...
```

Trident Protect 24.10.1 bis 25.06

Ab Trident Protect 24.10.1 friert Trident Protect KubeVirt-Dateisysteme während Datensicherungsoperationen automatisch ein und taut sie wieder auf. 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
```

Trident Protect 24.10

Trident Protect 24.10 stellt den konsistenten Zustand der KubeVirt VM-Dateisysteme während Datensicherungsoperationen nicht automatisch sicher. Wenn Sie Ihre KubeVirt VM-Daten mit Trident Protect 24.10 schützen möchten, müssen Sie die Funktion zum Einfrieren/Auftauen der Dateisysteme vor der Datensicherungsoperation 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 Auftauen des VM-Dateisystems während Datensicherungsoperationen verwaltet wird, indem "Virtualisierung konfigurieren" und anschließend den folgenden Befehl verwenden:

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

Anforderungen für die SnapMirror-Replikation

NetApp SnapMirror Replikation ist für die Verwendung mit Trident Protect für die folgenden ONTAP Lösungen verfügbar:

- Lokale NetApp FAS, AFF und ASA Systeme. SnapMirror-Replikation mit Trident protect wird derzeit für ASA r2 Systeme nicht unterstützt.
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSx for NetApp ONTAP

ONTAP Clusteranforderungen für SnapMirror Replikation

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

- **NetApp Trident**: NetApp Trident muss sowohl auf dem Quell- als auch auf dem Ziel-Kubernetes-Cluster vorhanden sein, die ONTAP als Backend verwenden. Trident Protect unterstützt die Replikation mit NetApp SnapMirror-Technologie unter Verwendung von Speicherklassen, die von den folgenden Treibern unterstützt werden:
  - `ontap-nas`: NFS
  - `ontap-san`: iSCSI
  - `ontap-san`: FC
  - `ontap-san`: NVMe/TCP (erfordert mindestens ONTAP Version 9.15.1)
- **Lizenzen**: ONTAP SnapMirror asynchrone Lizenzen mit dem Data Protection Bundle müssen sowohl auf dem Quell- als auch auf dem Ziel-ONTAP-Cluster aktiviert sein. Weitere Informationen sind unter "SnapMirror Lizenzierungsübersicht in ONTAP" verfügbar.

  Ab ONTAP 9.10.1 werden alle Lizenzen als NetApp Lizenzdatei (NLF) bereitgestellt, bei der es sich um eine einzelne Datei handelt, die mehrere Funktionen aktiviert. Weitere Informationen sind unter "In ONTAP One enthaltene Lizenzen" verfügbar.

Es wird ausschließlich SnapMirror asynchroner Schutz unterstützt.

Peering-Überlegungen für die SnapMirror Replikation

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

- **Cluster und SVM**: Die ONTAP-Speicher-Backends müssen per Peering verbunden sein. Weitere Informationen sind unter "Cluster- und SVM-Peering-Übersicht" verfügbar.

  Stellen Sie sicher, dass die in der Replikationsbeziehung zwischen zwei ONTAP Clustern verwendeten SVM-Namen eindeutig sind.

- **NetApp Trident und SVM**: Die verbundenen Remote-SVMs müssen für NetApp Trident auf dem Ziel-Cluster verfügbar sein.
- **Verwaltete Backends**: Sie müssen ONTAP-Speicher-Backends in Trident Protect hinzufügen und verwalten, um eine Replikationsbeziehung zu erstellen.

Trident / ONTAP-Konfiguration für SnapMirror-Replikation

Trident Protect erfordert, dass Sie mindestens ein Storage-Backend konfigurieren, das die Replikation sowohl für den Quell- als auch für den Ziel-Cluster unterstützt. Wenn der Quell- und der Ziel-Cluster identisch sind, sollte die Zielanwendung für die beste Ausfallsicherheit ein anderes Storage-Backend als die Quellanwendung verwenden.

Kubernetes-Cluster-Anforderungen für SnapMirror Replikation

Stellen Sie sicher, dass Ihre Kubernetes-Cluster die folgenden Anforderungen erfüllen:

- **AppVault-Zugänglichkeit**: Sowohl Quell- als auch Ziel-Cluster müssen Netzwerkzugriff haben, um vom AppVault zu lesen und darauf zu schreiben, damit Anwendungsobjekte repliziert werden können.
- **Netzwerkanbindung**: Konfigurieren Sie Firewall-Regeln, Bucket-Berechtigungen und IP-Zulassungslisten, um die Kommunikation zwischen beiden Clustern und dem AppVault über WANs hinweg zu ermöglichen.

Viele Unternehmensumgebungen setzen strenge Firewall-Richtlinien für WAN-Verbindungen ein. Klären Sie diese Netzwerkanforderungen mit Ihrem Infrastrukturteam, bevor Sie die Replikation konfigurieren.

Trident Protect installieren und konfigurieren

Wenn Ihre Umgebung die Anforderungen für Trident Protect erfüllt, können Sie Trident Protect mithilfe der folgenden Schritte auf Ihrem Cluster installieren. Sie können Trident Protect von NetApp beziehen oder es aus Ihrer eigenen privaten Registry installieren. Die Installation aus einer privaten Registry ist hilfreich, wenn Ihr Cluster keinen Internetzugang hat.

Installieren Sie Trident Protect

Installieren Sie Trident Protect von NetApp

Schritte

1. Fügen Sie das Trident Helm repository hinzu:

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

2. Verwenden Sie Helm, um Trident Protect zu installieren. Ersetzen Sie `<name-of-cluster>` durch einen Clusternamen, der dem Cluster zugewiesen und zur Identifizierung der Backups und Snapshots des Clusters verwendet wird:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2602.0 --create
-namespace --namespace trident-protect
```

3. Optional können Sie zur Aktivierung der Debug-Protokollierung (empfohlen zur Fehlerbehebung) Folgendes verwenden:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2602.0 --create-namespace --namespace trident-protect
```

   Die Debug-Protokollierung hilft dem NetApp Support, Probleme zu beheben, ohne dass Änderungen des Protokollierungslevels oder eine Reproduktion des Problems erforderlich sind.

Installieren Sie Trident Protect aus einem privaten Registry.

Sie können Trident Protect aus einer privaten Image-Registry installieren, wenn Ihr Kubernetes-Cluster keinen Internetzugang hat. Ersetzen Sie in diesen Beispielen die Werte in Klammern durch Informationen aus Ihrer Umgebung:

Schritte

1. Laden Sie die folgenden Images auf Ihre lokale Maschine herunter, aktualisieren Sie die Tags und laden Sie sie anschließend in Ihre private Registry hoch:

```
docker.io/netapp/controller:26.02.0
docker.io/netapp/restic:26.02.0
docker.io/netapp/kopia:26.02.0
docker.io/netapp/kopiablockrestore:26.02.0
docker.io/netapp/trident-autosupport:26.02.0
docker.io/netapp/exechook:26.02.0
docker.io/netapp/resourcebackup:26.02.0
docker.io/netapp/resourcerestore:26.02.0
docker.io/netapp/resourcedelete:26.02.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

Beispiel:

```
docker pull docker.io/netapp/controller:26.02.0
```

```
docker tag docker.io/netapp/controller:26.02.0 <private-registry-url>/controller:26.02.0
```

```
docker push <private-registry-url>/controller:26.02.0
```

> Um das Helm-Chart zu erhalten, laden Sie zunächst das Helm-Chart auf einem Rechner mit Internetzugang mit `helm pull trident-protect --version 100.2602.0 --repo https://netapp.github.io/trident-protect-helm-chart` herunter, kopieren Sie dann die resultierende `trident-protect-100.2602.0.tgz` Datei in Ihre Offline-Umgebung und installieren Sie es mit `helm install trident-protect ./trident-protect-100.2602.0.tgz` anstelle der Repository-Referenz im letzten Schritt.

2. Erstellen Sie den Trident Protect-Systemnamensraum:

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

3. Melden Sie sich bei der Registry an:

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

4. Erstellen Sie ein Pull-Secret zur Verwendung für die private Registry-Authentifizierung:

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

5. Fügen Sie das Trident Helm repository hinzu:

```
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 sie die folgenden Trident Protect-Einstellungen enthält:

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

> Die `imageRegistry` und `imagePullSecrets` Werte gelten für alle Komponentenbilder, einschließlich `resourcebackup` und `resourcerestore`. Wenn Sie Bilder in einen bestimmten Repository-Pfad innerhalb Ihrer Registry hochladen (zum Beispiel `example.com:443/my-repo`), geben Sie den vollständigen Pfad im Registry-Feld an. Dadurch wird sichergestellt, dass alle Bilder aus `<private-registry-url>/<image-name>:<tag>` abgerufen werden.

7. Verwenden Sie Helm, um Trident Protect zu installieren. Ersetzen Sie `<name_of_cluster>` durch einen Clusternamen, der dem Cluster zugewiesen und zur Identifizierung der Backups und Snapshots des Clusters verwendet wird:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name_of_cluster> --version 100.2602.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

8. Optional können Sie zur Aktivierung der Debug-Protokollierung (empfohlen zur Fehlerbehebung) Folgendes verwenden:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2602.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

Die Debug-Protokollierung hilft dem NetApp Support, Probleme zu beheben, ohne dass Änderungen des Protokollierungslevels oder eine Reproduktion des Problems erforderlich sind.

Weitere Helm-Chart-Konfigurationsoptionen, einschließlich AutoSupport-Einstellungen und Namespace-Filterung, finden Sie unter "Trident Protect-Installation anpassen".

Installieren Sie das Trident Protect CLI-Plugin

Sie können das Trident Protect Befehlszeilen-Plugin verwenden, eine Erweiterung des Trident `tridentctl`-Dienstprogramms, um Trident Protect benutzerdefinierte Ressourcen (CRs) zu erstellen und mit ihnen zu interagieren.

Installieren Sie das Trident Protect CLI-Plugin

Bevor Sie das Befehlszeilenprogramm verwenden, müssen Sie es auf dem Rechner installieren, mit dem Sie auf Ihren Cluster zugreifen. Führen Sie die folgenden Schritte aus, je nachdem, ob Ihr Rechner eine x64- oder ARM-CPU verwendet.

Plugin für Linux AMD64-CPUs herunterladen

Schritte

1. Laden Sie das Trident Protect CLI-Plugin herunter:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-linux-amd64
```

Plugin für Linux ARM64 CPUs herunterladen

Schritte

1. Laden Sie das Trident Protect CLI-Plugin herunter:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-linux-arm64
```

Plugin für Mac AMD64-CPUs herunterladen

Schritte

1. Laden Sie das Trident Protect CLI-Plugin herunter:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-macos-amd64
```

Plugin für Mac ARM64 CPUs herunterladen

Schritte

1. Laden Sie das Trident Protect CLI-Plugin herunter:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-macos-arm64
```

1. Aktivieren Sie die Ausführungsberechtigungen für die Plugin-Binärdatei:

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

2. Kopieren Sie die Plugin-Binärdatei in ein Verzeichnis, das in Ihrer PATH-Variablen definiert ist. Zum Beispiel, `/usr/bin` oder `/usr/local/bin` (möglicherweise benötigen Sie erhöhte Berechtigungen):

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

3. Optional können Sie die Plugin-Binärdatei in ein Verzeichnis in Ihrem Home-Verzeichnis kopieren. In diesem Fall wird empfohlen, sicherzustellen, dass sich das Verzeichnis in Ihrer PATH-Variablen befindet:

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

Durch das Kopieren des Plugins an einen Ort in Ihrer PATH-Variablen können Sie das Plugin verwenden, indem Sie `tridentctl-protect` oder `tridentctl protect` von jedem beliebigen Ort aus eingeben.

Hilfe zum Trident CLI-Plugin anzeigen

Sie können die integrierten Plugin-Hilfefunktionen verwenden, um detaillierte Hilfe zu den Funktionen des Plugins zu erhalten:

Schritte

1. Verwenden Sie die Hilfefunktion, um Nutzungshinweise anzuzeigen:

```
tridentctl-protect help
```

Automatische Befehlsvervollständigung aktivieren

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. Erstellen Sie das Vervollständigungsskript:

```
tridentctl-protect completion bash > tridentctl-completion.bash
```

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

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

3. Verschieben Sie das heruntergeladene Skript in das `~/.bash/completions` Verzeichnis:

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

4. Fügen Sie die folgende Zeile in die `~/.bashrc` Datei in Ihrem Home-Verzeichnis ein:

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

Automatische Vervollständigung für die Z shell aktivieren

Schritte

1. Erstellen Sie das Vervollständigungsskript:

```
tridentctl-protect completion zsh > tridentctl-completion.zsh
```

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

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

3. Verschieben Sie das heruntergeladene Skript in das `~/.zsh/completions` Verzeichnis:

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

4. Fügen Sie die folgende Zeile in die `~/.zprofile` Datei in Ihrem Home-Verzeichnis ein:

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

Ergebnis

Beim nächsten Login in die Shell können Sie die Befehlsvervollständigung mit dem tridentctl-protect plugin nutzen.

Trident Protect-Installation anpassen

Sie können die Standardkonfiguration von Trident Protect an die spezifischen Anforderungen Ihrer Umgebung anpassen.

Geben Sie die Ressourcenbeschränkungen für den Trident Protect-Container an.

Sie können eine Konfigurationsdatei verwenden, um Ressourcenlimits für Trident Protect-Container festzulegen, nachdem Sie Trident Protect installiert haben. Durch das Festlegen von Ressourcenlimits können Sie steuern, wie viele Ressourcen des Clusters von Trident Protect-Operationen verbraucht werden.

Schritte

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

   Die folgende Beispiel-Konfigurationsdatei zeigt die verfügbaren Einstellungen und enthält die Standardwerte für jede Ressourcenbegrenzung:

```
---
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. Wenden Sie die Werte aus der `resourceLimits.yaml` Datei an:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

Sicherheitskontextbeschränkungen anpassen

Sie können eine Konfigurationsdatei verwenden, um die OpenShift Security Context Constraint (SCCs) für Trident Protect-Container nach der Installation von Trident Protect zu ändern. Diese Beschränkungen definieren Sicherheitsvorgaben für Pods in einem Red Hat OpenShift Cluster.

Schritte

1. Erstellen Sie eine Datei mit dem Namen `sccconfig.yaml`.
2. Fügen Sie die SCC-Option zur Datei hinzu und ändern Sie die Parameter entsprechend den Anforderungen Ihrer Umgebung.

   Das folgende Beispiel zeigt die Standardwerte der Parameter für die SCC-Option:

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

Diese Tabelle beschreibt die Parameter für die SCC-Option:

Parameter	Beschreibung	Standard
erstellen	Legt fest, ob eine SCC-Ressource erstellt werden kann. Eine SCC-Ressource wird nur erstellt, wenn `scc.create` auf `true` gesetzt ist und der Helm-Installationsprozess eine OpenShift-Umgebung identifiziert. Wenn nicht auf OpenShift gearbeitet wird oder wenn `scc.create` auf `false` gesetzt ist, wird keine SCC-Ressource erstellt.	true
Name	Gibt den Namen des SCC an.	trident-protect-job
Priorität	Legt die Priorität der SCC fest. SCCs mit höheren Prioritätswerten werden vor solchen mit niedrigeren Werten bewertet.	1

3. Wenden Sie die Werte aus der `sccconfig.yaml` Datei an:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

Dadurch werden die Standardwerte durch die in der `sccconfig.yaml` Datei angegebenen Werte ersetzt.

Konfigurieren Sie zusätzliche Trident Protect Helm-Chart-Einstellungen

Sie können die AutoSupport-Einstellungen und die Namensraumfilterung an Ihre spezifischen Anforderungen anpassen. Die folgende Tabelle beschreibt die verfügbaren Konfigurationsparameter:

Parameter	Typ	Beschreibung
autoSupport.proxy	Zeichenkette	Konfiguriert eine Proxy-URL für NetApp AutoSupport-Verbindungen. Verwenden Sie dies, um Support-Bundle-Uploads über einen Proxyserver zu leiten. Beispiel: `http://my.proxy.url`.

Parameter	Typ	Beschreibung
autoSupport.insecure	boolescher Wert	Überspringt die TLS-Verifizierung für AutoSupport Proxyserver-Verbindungen, wenn auf `true` gesetzt. Nur für unsichere Proxyserver-Verbindungen verwenden. (Standard: `false`)
autoSupport.enabled	boolescher Wert	Aktiviert oder deaktiviert tägliche Trident Protect AutoSupport Bundle-Uploads. Wenn auf `false` gesetzt, werden geplante tägliche Uploads deaktiviert, aber Sie können weiterhin Support-Bundles manuell generieren. (Standard: `true`)
restoreSkipNamespaceAnnotations	Zeichenkette	Kommagetrennte Liste von Namespace-Annotationen, die von Sicherungs- und Wiederherstellungsvorgängen ausgeschlossen werden sollen. Ermöglicht das Filtern von Namespaces anhand von Annotationen.
restoreSkipNamespaceLabels	Zeichenkette	Durch Kommas getrennte Liste von Namespace-Labels, die von Sicherungs- und Wiederherstellungsvorgängen ausgeschlossen werden. Ermöglicht das Filtern von Namespaces anhand von Labels.

Sie können diese Optionen entweder mit einer YAML-Konfigurationsdatei oder Befehlszeilen-Flags konfigurieren:

YAML-Datei verwenden

Schritte

1. Erstelle eine Konfigurationsdatei und benenne sie `values.yaml`.
2. Fügen Sie in der von Ihnen erstellten Datei die Konfigurationsoptionen hinzu, die Sie anpassen möchten.

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
```

3. Nachdem Sie die `values.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die Konfigurationsdatei an:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

CLI-Flag verwenden

Schritte

1. Verwenden Sie den folgenden Befehl mit dem `--set` Flag, um einzelne Parameter anzugeben:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set-string
restoreSkipNamespaceAnnotations="{annotation1,annotation2}" \
  --set-string restoreSkipNamespaceLabels="{label1,label2}" \
  --reuse-values
```

Trident Protect-Pods auf bestimmte Knoten beschränken

Sie können die Kubernetes nodeSelector Knotenauswahlbeschränkung verwenden, um anhand von Knotenbezeichnungen zu steuern, welche Ihrer Knoten berechtigt sind, Trident Protect-Pods auszuführen. Standardmäßig ist Trident Protect auf Knoten beschränkt, auf denen Linux ausgeführt wird. Sie können diese Beschränkungen je nach Bedarf weiter anpassen.

Schritte

1. Erstellen Sie eine Datei mit dem Namen `nodeSelectorConfig.yaml`.

2. Fügen Sie die nodeSelector-Option zur Datei hinzu und bearbeiten Sie die Datei, um Knotenbezeichnungen hinzuzufügen oder zu ändern, um die Einschränkungen entsprechend den Anforderungen Ihrer Umgebung anzupassen. Die folgende Datei enthält beispielsweise die Standardeinschränkung des Betriebssystems, zielt aber auch auf eine bestimmte Region und einen bestimmten Anwendungsnamen ab:

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. Wenden Sie die Werte aus der `nodeSelectorConfig.yaml` Datei an:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

Dadurch werden die Standardbeschränkungen durch die von Ihnen in der `nodeSelectorConfig.yaml` Datei angegebenen Beschränkungen ersetzt.

Trident Protect verwalten

Trident Protect-Autorisierung und Zugriffskontrolle verwalten

Trident Protect nutzt das Kubernetes-Modell der rollenbasierten Zugriffssteuerung (RBAC). Standardmäßig stellt Trident Protect einen einzelnen System-Namespace und das zugehörige Standard-Dienstkonto bereit. Wenn Sie eine Organisation mit vielen Benutzern oder spezifischen Sicherheitsanforderungen haben, können Sie die RBAC-Funktionen von Trident Protect nutzen, um den Zugriff auf Ressourcen und Namespaces detaillierter zu steuern.

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

Beachten Sie, dass keine Benutzer Anwendungsdatenverwaltungs-CRs im Standard `trident-protect` -Namespace erstellen können. Sie müssen Anwendungsdatenverwaltungs-CRs in einem Anwendungs-Namespace erstellen (als Best Practice erstellen Sie Anwendungsdatenverwaltungs-CRs im selben Namespace wie die zugehörige Anwendung).

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

- **AppVault**: Erfordert Bucket-Anmeldeinformationen
- **AutoSupportBundle**: Erfasst Metriken, Protokolle und andere sensible Trident Protect-Daten
- **AutoSupportBundleSchedule**: Verwaltet Protokollerfassungspläne

Als bewährte Methode verwenden Sie rollenbasierte Zugriffssteuerung (RBAC), um den Zugriff auf privilegierte Objekte auf Administratoren zu beschränken.

Weitere Informationen darüber, wie RBAC den Zugriff auf Ressourcen und Namensräume regelt, finden Sie unter "Kubernetes RBAC-Dokumentation".

Weitere Informationen zu Servicekonten finden Sie in der "Dokumentation zum Kubernetes Servicekonto".

Beispiel: Zugriff für zwei Gruppen von Benutzern verwalten

Eine Organisation verfügt beispielsweise über einen Cluster-Administrator, eine Gruppe von Engineering-Benutzern und eine Gruppe von Marketing-Benutzern. Der Cluster-Administrator würde die folgenden Aufgaben ausführen, um eine Umgebung zu schaffen, in der die Engineering-Gruppe und die Marketing-Gruppe jeweils nur auf die Ressourcen zugreifen können, die ihren jeweiligen Namensräumen zugewiesen sind.

Schritt 1: Erstellen Sie einen Namensraum, um Ressourcen für jede Gruppe zu enthalten

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

Schritte

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

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

2. Erstellen Sie einen Namensraum für die Marketinggruppe:

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

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

Jeder neu erstellte Namespace verfügt über ein Standarddienstkonto, aber Sie sollten für jede Benutzergruppe ein Dienstkonto erstellen, damit Sie die Berechtigungen in Zukunft bei Bedarf weiter zwischen den Gruppen aufteilen können.

Schritte

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

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

2. Erstellen Sie ein Servicekonto für die Marketinggruppe:

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

Schritt 3: Erstellen Sie ein Geheimnis für jedes neue Dienstkonto

Ein Dienstkontogeheimnis wird zur Authentifizierung mit dem Dienstkonto verwendet und kann im Falle einer Kompromittierung leicht gelöscht und neu erstellt werden.

Schritte

1. Erstellen Sie ein Secret für das Engineering-Service-Konto:

```
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 Marketing-Service-Konto:

```
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 Dienstkonto zu binden

Ein Standard-ClusterRole-Objekt wird erstellt, wenn Sie Trident Protect installieren. Sie können dieses 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 die ClusterRole an das Marketing-Servicekonto:

```
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: Berechtigungen testen

Testen Sie, ob die Berechtigungen korrekt sind.

Schritte

1. Bestätigen Sie, dass technische Benutzer auf technische 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 technische Benutzer keinen Zugriff auf Marketingressourcen haben:

```
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 Datenverwaltungsaufgaben wie Backups und Snapshots durchzuführen, muss der Clusteradministrator einzelnen Benutzern Zugriff auf AppVault-Objekte gewähren.

Schritte

1. Erstellen und wenden Sie eine AppVault- und Secret-Kombinations-YAML-Datei an, die einem Benutzer Zugriff auf eine AppVault gewährt. Beispielsweise gewährt die folgende CR einem Benutzer Zugriff auf eine 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 wenden Sie eine Role-CR an, um Clusteradministratoren die Möglichkeit zu geben, Zugriff auf bestimmte Ressourcen in einem Namespace zu gewähren. 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 eine 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. Versuch, AppVault-Objektinformationen für alle Namensräume abzurufen:

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

   Sie sollten eine Ausgabe ähnlich der folgenden 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 abrufen kann, auf die er nun Zugriffsberechtigung 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 ähnlich der folgenden sehen:

```
yes
```

Ergebnis

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

Überwachen Sie die Trident Protect-Ressourcen

Sie können die Open-Source-Tools kube-state-metrics, Prometheus und Alertmanager verwenden, um die Integrität der durch Trident Protect geschützten Ressourcen zu überwachen.

Der Dienst kube-state-metrics generiert Metriken aus der Kubernetes-API-Kommunikation. Die Verwendung mit Trident Protect liefert nützliche Informationen über den Zustand der Ressourcen in Ihrer Umgebung.

Prometheus ist ein Toolkit, das die von kube-state-metrics generierten Daten erfassen und als leicht lesbare Informationen über diese Objekte darstellen kann. Zusammen bieten kube-state-metrics und Prometheus eine Möglichkeit, den Zustand und Status der Ressourcen zu überwachen, die Sie mit Trident Protect verwalten.

Alertmanager ist ein Dienst, der die von Tools wie Prometheus gesendeten Warnmeldungen aufnimmt und sie an Ziele weiterleitet, die Sie konfigurieren.

> Die in diesen Schritten enthaltenen Konfigurationen und Anleitungen sind lediglich Beispiele; Sie müssen sie an Ihre Umgebung anpassen. Spezifische Anweisungen und Unterstützung finden Sie in der folgenden offiziellen Dokumentation:
>
> - "kube-state-metrics Dokumentation"
> - "Prometheus-Dokumentation"
> - "Alertmanager-Dokumentation"

Schritt 1: Installieren Sie die Überwachungstools

Um die Ressourcenüberwachung in Trident Protect zu aktivieren, müssen Sie kube-state-metrics, Promethus und Alertmanager installieren und konfigurieren.

Installieren Sie kube-state-metrics

Sie können kube-state-metrics mit Helm installieren.

Schritte

1. Fügen Sie das kube-state-metrics Helm-Chart hinzu. Beispiel:

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update
```

2. Wenden Sie die Prometheus ServiceMonitor CRD auf den Cluster an:

```
kubectl apply -f https://raw.githubusercontent.com/prometheus-
operator/prometheus-operator/main/example/prometheus-operator-
crd/monitoring.coreos.com_servicemonitors.yaml
```

3. Erstellen Sie eine Konfigurationsdatei für das Helm-Chart (zum Beispiel `metrics-config.yaml`). Sie können die folgende Beispielkonfiguration an Ihre Umgebung anpassen:

metrics-config.yaml: kube-state-metrics Helm-Chart-Konfiguration

```
---
extraArgs:
  # Collect only custom metrics
  - --custom-resource-state-only=true

customResourceState:
  enabled: true
  config:
    kind: CustomResourceStateMetrics
    spec:
      resources:
      - groupVersionKind:
          group: protect.trident.netapp.io
          kind: "Backup"
          version: "v1"
        labelsFromPath:
          backup_uid: [metadata, uid]
          backup_name: [metadata, name]
          creation_time: [metadata, creationTimestamp]
        metrics:
        - name: backup_info
          help: "Exposes details about the Backup state"
          each:
            type: Info
            info:
              labelsFromPath:
                appVaultReference: ["spec", "appVaultRef"]
                appReference: ["spec", "applicationRef"]
rbac:
  extraRules:
  - apiGroups: ["protect.trident.netapp.io"]
    resources: ["backups"]
    verbs: ["list", "watch"]

# Collect metrics from all namespaces
namespaces: ""

# Ensure that the metrics are collected by Prometheus
prometheus:
  monitor:
    enabled: true
```

4. Installieren Sie kube-state-metrics, indem Sie das Helm-Chart bereitstellen. Beispiel:

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. Konfigurieren Sie kube-state-metrics, um Metriken für die von Trident Protect verwendeten benutzerdefinierten Ressourcen zu generieren, indem Sie den Anweisungen in der "kube-state-metrics Custom Resource Dokumentation" folgen.

Prometheus installieren

Sie können Prometheus installieren, indem Sie den Anweisungen in der "Prometheus-Dokumentation" folgen.

Installieren Sie Alertmanager

Sie können Alertmanager installieren, indem Sie den Anweisungen in der "Alertmanager-Dokumentation" folgen.

Schritt 2: Konfigurieren Sie die Überwachungstools so, dass sie zusammenarbeiten

Nach der Installation der Überwachungstools müssen Sie diese so konfigurieren, dass sie zusammenarbeiten.

Schritte

1. Integrieren Sie kube-state-metrics in Prometheus. Bearbeiten Sie die Prometheus-Konfigurationsdatei (`prometheus.yaml` und fügen Sie die Informationen zum kube-state-metrics-Service hinzu. Beispiel:

   prometheus.yaml: Integration des kube-state-metrics service mit Prometheus

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: trident-protect
data:
  prometheus.yaml: |
    global:
      scrape_interval: 15s
    scrape_configs:
      - job_name: 'kube-state-metrics'
        static_configs:
          - targets: ['kube-state-metrics.trident-protect.svc:8080']
```

2. Konfigurieren Sie Prometheus so, dass Warnmeldungen an Alertmanager weitergeleitet werden. Bearbeiten Sie die Prometheus Konfigurationsdatei (`prometheus.yaml` und fügen Sie den folgenden Abschnitt hinzu:

prometheus.yaml: Senden Sie Warnmeldungen an Alertmanager

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager.trident-protect.svc:9093
```

Ergebnis

Prometheus kann nun Metriken von kube-state-metrics erfassen und Warnmeldungen an Alertmanager senden. Sie sind jetzt bereit zu konfigurieren, welche Bedingungen eine Warnmeldung auslösen und wohin die Warnmeldungen gesendet werden sollen.

Schritt 3: Benachrichtigungen und Benachrichtigungsziele konfigurieren

Nachdem Sie die Tools so konfiguriert haben, dass sie zusammenarbeiten, müssen Sie konfigurieren, welche Art von Informationen Warnmeldungen auslösen und wohin die Warnmeldungen gesendet werden sollen.

Warnungsbeispiel: Backup-Fehler

Das folgende Beispiel definiert eine kritische Warnung, die ausgelöst wird, wenn der Status der benutzerdefinierten Backup-Ressource auf `Error` für 5 Sekunden oder länger gesetzt ist. Sie können dieses Beispiel an Ihre Umgebung anpassen und diesen YAML-Ausschnitt in Ihre `prometheus.yaml` Konfigurationsdatei einfügen:

rules.yaml: Definiere eine Prometheus-Alarmierung für fehlgeschlagene Backups

```
rules.yaml: |
  groups:
    - name: fail-backup
        rules:
          - alert: BackupFailed
            expr: kube_customresource_backup_info{status="Error"}
            for: 5s
            labels:
              severity: critical
            annotations:
              summary: "Backup failed"
              description: "A backup has failed."
```

Konfigurieren Sie Alertmanager, um Benachrichtigungen an andere Kanäle zu senden

Sie können Alertmanager so konfigurieren, dass Benachrichtigungen an andere Kanäle wie E-Mail, PagerDuty, Microsoft Teams oder andere Benachrichtigungsdienste gesendet werden, indem Sie die jeweilige Konfiguration in der `alertmanager.yaml` Datei angeben.

Das folgende Beispiel konfiguriert Alertmanager so, dass Benachrichtigungen an einen Slack-Kanal gesendet werden. Um dieses Beispiel an Ihre Umgebung anzupassen, ersetzen Sie den Wert des `api_url` Schlüssels durch die in Ihrer Umgebung verwendete Slack-Webhook-URL:

alertmanager.yaml: Senden Sie Warnmeldungen an einen Slack-Kanal

```
data:
  alertmanager.yaml: |
    global:
      resolve_timeout: 5m
    route:
      receiver: 'slack-notifications'
    receivers:
      - name: 'slack-notifications'
        slack_configs:
          - api_url: '<your-slack-webhook-url>'
            channel: '#failed-backups-channel'
            send_resolved: false
```

Generieren Sie ein Trident Protect Support-Bundle

Trident Protect ermöglicht Administratoren, Bundles zu erstellen, die Informationen enthalten, die für den NetApp Support nützlich sind, einschließlich Protokollen, Metriken und Topologieinformationen über die verwalteten Cluster und Apps. Wenn Sie mit dem Internet verbunden sind, können Sie Support-Bundles mithilfe einer benutzerdefinierten Ressourcendatei (CR) auf die NetApp Support Site (NSS) hochladen.

Erstellen Sie ein Support-Bundle mithilfe eines CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (zum Beispiel `trident-protect-support-bundle.yaml`).
2. Konfigurieren Sie die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.triggerType**: (*Erforderlich*) Legt fest, ob das Support-Bundle sofort generiert oder geplant wird. Die geplante Bundle-Generierung erfolgt um 12:00 Uhr UTC. Mögliche Werte:
     - Geplant
     - Handbuch
   - **spec.uploadEnabled**: (*Optional*) Steuert, ob das Support-Bundle nach seiner Generierung auf die NetApp Support-Website hochgeladen werden soll. Wenn nicht angegeben, ist der Standardwert `false`. Mögliche Werte:
     - true
     - false (Standardeinstellung)
   - **spec.dataWindowStart**: (*Optional*) Eine Datumszeichenkette 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, wird standardmäßig 24 Stunden zurück angenommen. Das früheste Fensterdatum, das Sie angeben können, liegt 7 Tage zurück.

     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 `trident-protect-support-bundle.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

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

Erstellen Sie ein Support-Bundle mithilfe der CLI

Schritte

1. Erstellen Sie das Support-Bundle, indem Sie die Werte in Klammern durch Informationen aus Ihrer

Umgebung ersetzen. Das `trigger-type` bestimmt, ob das Bundle sofort erstellt wird oder ob die Erstellungszeit durch den Zeitplan vorgegeben ist, und kann `Manual` oder `Scheduled` sein. Die Standardeinstellung ist `Manual`.

Beispiel:

```
tridentctl-protect create autosupportbundle <my-bundle-name>
--trigger-type <trigger-type> -n trident-protect
```

Überwachen und rufen Sie das Support-Paket ab

Nachdem Sie mit einer der beiden Methoden ein Support-Bundle erstellt haben, können Sie den Generierungsfortschritt überwachen und es auf Ihr lokales System abrufen.

Schritte

1. Warten Sie, bis der `status.generationState` den `Completed` Status erreicht. Sie können den Generierungsfortschritt mit dem folgenden Befehl überwachen:

   ```
   kubectl get autosupportbundle trident-protect-support-bundle -n trident-
   protect
   ```

2. Rufen Sie das Support-Bundle auf Ihr lokales System ab. Rufen Sie den Kopierbefehl aus dem abgeschlossenen AutoSupport-Bundle ab:

   ```
   kubectl describe autosupportbundle trident-protect-support-bundle -n
   trident-protect
   ```

   Suchen Sie den `kubectl cp` Befehl in der Ausgabe und führen Sie ihn aus, wobei Sie das Argument Ziel durch Ihr bevorzugtes lokales Verzeichnis ersetzen.

Trident Protect aktualisieren

Sie können Trident Protect auf die neueste Version aktualisieren, um neue Funktionen oder Fehlerbehebungen zu nutzen.

- Beim Upgrade von Version 24.10 kann es vorkommen, dass während des Upgrades laufende Snapshots fehlschlagen. Dieses Fehlschlagen verhindert nicht, dass zukünftige Snapshots, egal ob manuell oder geplant, erstellt werden können. Wenn ein Snapshot während des Upgrades fehlschlägt, können Sie manuell einen neuen Snapshot erstellen, um sicherzustellen, dass Ihre Anwendung geschützt ist.

  Um mögliche Fehler zu vermeiden, können Sie vor dem Upgrade alle Snapshot-Zeitpläne deaktivieren und sie danach wieder aktivieren. Dies führt jedoch dazu, dass während des Upgrade-Zeitraums geplante Snapshots fehlen.

- Bei Installationen in privaten Registrys stellen Sie sicher, dass das erforderliche Helm-Chart und die Images für die Zielversion in Ihrer privaten Registry verfügbar sind, und überprüfen Sie, ob Ihre benutzerdefinierten Helm-Werte mit der neuen Chart-Version kompatibel sind. Weitere Informationen finden sich unter "Installieren Sie Trident Protect aus einem privaten Registry.".

Schritt 1: Wählen Sie eine Version

Trident Protect-Versionen folgen einer datumsbasierten `YY.MM` Namenskonvention, wobei „YY“ die letzten beiden Ziffern des Jahres und „MM“ der Monat sind. Dot-Releases folgen einer `YY.MM.X` Konvention, wobei „X“ das Patch-Level ist. Sie wählen die Version aus, auf die Sie aktualisieren möchten, basierend auf der Version, von der Sie aktualisieren.

- Sie können ein direktes Upgrade auf jede Zielversion durchführen, die innerhalb eines Vier-Versions-Fensters Ihrer installierten Version liegt. Beispielsweise können Sie direkt von 24.10 (oder jeder 24.10 dot release) auf 25.10 aktualisieren.
- Wenn Sie von einer Version außerhalb des Vier-Versions-Fensters aktualisieren, ist ein mehrstufiges Upgrade erforderlich. Die Upgrade-Anweisungen für die "frühere Version" Version, von der Sie aktualisieren, sind zu verwenden, um auf die neueste Version innerhalb des Vier-Versions-Fensters zu aktualisieren. Beispielsweise, wenn Version 24.10 verwendet wird und ein Upgrade auf 26.02 erfolgen soll:
  a. Erstes Upgrade von 24.10 auf 25.02.
  b. Führen Sie dann ein Upgrade von 25.02 auf 26.02 durch.

Schritt 2: Trident Protect aktualisieren

Um Trident Protect zu aktualisieren, führen Sie die folgenden Schritte aus.

Schritte

1. Aktualisieren Sie das Trident Helm-Repository:

```
helm repo update
```

2. Aktualisieren Sie die Trident Protect CRDs:

   Dieser Schritt ist erforderlich, wenn Sie von einer Version vor 25.06 aktualisieren, da die CRDs jetzt im Trident Protect Helm chart enthalten sind.

   a. Führen Sie diesen Befehl aus, um die Verwaltung von CRDs von `trident-protect-crds` zu `trident-protect` zu verschieben:

```
kubectl get crd | grep protect.trident.netapp.io | awk '{print $1}' |
xargs -I {} kubectl patch crd {} --type merge -p '{"metadata":
{"annotations":{"meta.helm.sh/release-name": "trident-protect"}}}'
```

b. Führen Sie diesen Befehl aus, um das Helm-Secret für das `trident-protect-crds` Chart zu löschen:

> Deinstallieren Sie das `trident-protect-crds` Chart nicht mit Helm, da dies Ihre CRDs und alle zugehörigen Daten entfernen könnte.

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. Trident Protect aktualisieren:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2602.0 --namespace trident-protect
```

> Sie können den Protokollierungsgrad während des Upgrades konfigurieren, indem Sie `--set logLevel=debug` zum Upgrade-Befehl hinzufügen. Der Standard-Protokollierungsgrad ist `warn`. Debug-Protokollierung wird für die Fehlerbehebung empfohlen, da sie NetApp Support hilft, Probleme zu diagnostizieren, ohne dass Änderungen am Protokollierungsgrad oder eine Problemreproduktion erforderlich sind.

Anwendungen verwalten und schützen

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

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

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

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

Konfigurieren Sie die AppVault-Authentifizierung und Passwörter

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

Passwörter für das Data mover-Repository

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

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

Erstellen Sie ein Passwortgeheimnis für das Data Mover Repository

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

- Je nachdem, welchen Datenmover Sie verwenden, müssen Sie nur das entsprechende Passwort für diesen Datenmover angeben. Wenn Sie beispielsweise Restic verwenden und nicht planen, Kopia in Zukunft zu nutzen, können Sie beim Erstellen des Geheimnisses nur das Restic-Passwort angeben.
- Bewahren Sie das Passwort sicher auf. Sie benötigen es, um Daten auf demselben Cluster oder auf einem anderen Cluster wiederherzustellen. Wenn der Cluster oder das `trident-protect` Namespace gelöscht wird, können Sie Ihre Backups oder Snapshots ohne das Passwort nicht wiederherstellen.

Verwenden Sie einen CR

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

Verwenden Sie die Befehlszeile

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

S3-kompatible Speicher-IAM-Berechtigungen

Wenn Sie auf S3-kompatiblen Speicher wie Amazon S3, Generic S3, "StorageGrid S3", oder "ONTAP S3" mit Trident Protect zugreifen, müssen Sie sicherstellen, dass die von Ihnen bereitgestellten Benutzeranmeldeinformationen über die erforderlichen Berechtigungen für den Zugriff auf den Bucket verfügen. Im Folgenden finden Sie ein Beispiel für eine Richtlinie, die die minimal erforderlichen Berechtigungen für den Zugriff mit Trident Protect gewährt. Sie können diese Richtlinie auf den Benutzer anwenden, der die S3-kompatiblen Bucket-Richtlinien verwaltet.

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

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

EKS Pod Identity für Amazon S3 (AWS) Authentifizierung

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

Anforderungen für EKS Pod Identity mit Trident Protect

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

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

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

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

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

AppVault Schlüsselgenerierungsbeispiele für Cloud-Anbieter

Beim Definieren eines AppVault CR müssen Sie Anmeldeinformationen angeben, um auf die vom Anbieter gehosteten Ressourcen zuzugreifen, es sei denn, Sie verwenden IAM-Authentifizierung. Wie Sie die Schlüssel für die Anmeldeinformationen generieren, hängt vom jeweiligen Anbieter ab. Im Folgenden finden Sie Beispiele für die Schlüsselgenerierung über die Befehlszeile für verschiedene Anbieter. Sie können die folgenden Beispiele verwenden, um Schlüssel für die Anmeldeinformationen jedes Cloud-Anbieters zu erstellen.

Google Cloud

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

Amazon S3 (AWS)

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

Microsoft Azure

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

Generisches S3

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

ONTAP S3

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

StorageGrid S3

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

AppVault-Erstellungsbeispiele

Die folgenden sind Beispiel-AppVault-Definitionen für jeden Anbieter.

AppVault CR-Beispiele

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

- Optional können Sie ein Kubernetes-Secret angeben, das benutzerdefinierte Passwörter für die Verschlüsselung der Restic- und Kopia-Repositories enthält. Weitere Informationen sind unter Passwörter für das Data mover-Repository verfügbar.
- Für Amazon S3 (AWS) AppVault-Objekte können Sie optional ein sessionToken angeben, was nützlich ist, wenn Sie Single Sign-On (SSO) zur Authentifizierung verwenden. Dieses Token wird erstellt, wenn Sie Schlüssel für den Anbieter in AppVault Schlüsselgenerierungsbeispiele für Cloud-Anbieter generieren.
- Für S3 AppVault-Objekte können Sie optional eine Egress-Proxy-URL für ausgehenden S3-Datenverkehr mithilfe des `spec.providerConfig.S3.proxyURL` Schlüssels angeben.

Google Cloud

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

Amazon S3 (AWS)

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

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

Microsoft Azure

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

Generisches S3

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

ONTAP S3

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

StorageGrid S3

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

AppVault Erstellungsbeispiele mit der Trident Protect CLI

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

- Optional können Sie ein Kubernetes-Secret angeben, das benutzerdefinierte Passwörter für die Verschlüsselung der Restic- und Kopia-Repositories enthält. Weitere Informationen sind unter Passwörter für das Data mover-Repository verfügbar.
- Für S3-AppVault-Objekte können Sie optional eine Egress-Proxy-URL für ausgehenden S3-Datenverkehr mithilfe des `--proxy-url <ip_address:port>` Arguments angeben.

Google Cloud

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

Amazon S3 (AWS)

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

Microsoft Azure

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

Generisches S3

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

ONTAP S3

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

StorageGrid S3

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

Unterstützte `providerConfig.s3` Konfigurationsoptionen

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

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

Informationen zu AppVault anzeigen

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

Schritte

1. Den Inhalt eines AppVault Objekts anzeigen:

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

Beispielausgabe:

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

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

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

Entfernen Sie ein AppVault

Sie können ein AppVault-Objekt jederzeit entfernen.

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

Bevor Sie beginnen

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

Entfernen Sie ein AppVault mithilfe der Kubernetes-CLI

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

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

Entfernen Sie ein AppVault mit der Trident Protect CLI

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

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

Definieren Sie eine Anwendung für das Management mit Trident Protect

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

Erstellen Sie eine AppVault CR

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

Eine Anwendung definieren

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

Fügen Sie eine Anwendung mithilfe eines CR hinzu

Schritte

1. Erstellen Sie die CR-Datei der Zielanwendung:
   a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (zum Beispiel `maria-app.yaml`).
   b. Konfigurieren Sie die folgenden Attribute:
      - **metadata.name**: (*Erforderlich*) Der Name der benutzerdefinierten Anwendungsressource. Merken Sie sich den Namen, den Sie wählen, da andere für Schutzvorgänge benötigte CR-Dateien auf diesen Wert verweisen.
      - **spec.includedNamespaces**: (*Erforderlich*) Verwenden Sie Namespace und Label-Selektor, um die Namespaces und Ressourcen anzugeben, die die Anwendung verwendet. Der Anwendungsnamespace muss Teil dieser Liste sein. Der Label-Selektor ist optional und kann verwendet werden, um Ressourcen innerhalb jedes angegebenen Namespace zu filtern.
      - **spec.includedClusterScopedResources**: (*Optional*) Verwenden Sie dieses Attribut, um Cluster-Scoped-Ressourcen anzugeben, die in die Anwendungsdefinition aufgenommen werden sollen. Mit diesem Attribut können Sie diese Ressourcen anhand ihrer Gruppe, Version, Art und Bezeichnungen auswählen.
        - **groupVersionKind**: (*Erforderlich*) Gibt die API-Gruppe, die Version und die Art der clusterweiten Ressource an.
        - **labelSelector**: (*Optional*) Filtert die clusterweiten Ressourcen anhand ihrer Labels.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Optional*) Diese Annotation ist nur für Anwendungen relevant, die von virtuellen Maschinen aus definiert werden, z. B. in KubeVirt-Umgebungen, in denen das Dateisystem vor Snapshots eingefroren wird. Legen Sie fest, ob diese Anwendung während eines Snapshots auf das Dateisystem schreiben darf. Ist die Option auf true gesetzt, ignoriert die Anwendung die globale Einstellung und kann während eines Snapshots auf das Dateisystem schreiben. Ist die Option auf false gesetzt, ignoriert die Anwendung die globale Einstellung und das Dateisystem wird während eines Snapshots eingefroren. Wird die Option angegeben, die Anwendung aber keine virtuellen Maschinen in der Anwendungsdefinition hat, wird die Annotation ignoriert. Wird sie nicht angegeben, folgt die Anwendung der "globale Trident Protect freeze-Einstellung".

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

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

+
Beispiel YAML:

+

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

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

   - **Beispiel für einen generischen Filter:**

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

       - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese mit einer ODER-Verknüpfung verglichen und die Felder innerhalb jedes Elements (group, kind, version) werden mit einer UND-Verknüpfung verglichen.
       - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressource.

- **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
- **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
- **resourceMatchers[].names**: (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].labelSelectors**: (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

> Wenn sowohl `resourceFilter` als auch `labelSelector` verwendet werden, wird `resourceFilter` zuerst ausgeführt und anschließend `labelSelector` auf die resultierenden Ressourcen angewendet.

Beispiel:

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

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

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

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

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

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

Schritte

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

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

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

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

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

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

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

- Sie können `--resource-filter-include` und `--resource-filter-exclude` Flags verwenden, um Ressourcen basierend auf `resourceSelectionCriteria` wie Gruppe, Art, Version, Bezeichnungen, Namen und Namensräumen ein- oder auszuschließen, wie im folgenden Beispiel gezeigt:

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

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

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

Schützen Sie Anwendungen mit Trident Protect

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

erstellen.

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

Erstellen Sie einen On-Demand-Snapshot

Sie können jederzeit einen On-Demand-Snapshot erstellen.

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

Erstellen Sie einen Snapshot mithilfe eines CR

Schritte

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

     ```
     apiVersion: protect.trident.netapp.io/v1
     kind: Snapshot
     metadata:
       namespace: my-app-namespace
       name: my-cr-name
     spec:
       applicationRef: my-application
       appVaultRef: appvault-name
       reclaimPolicy: Delete
     ```
3. Nachdem Sie die `trident-protect-snapshot-cr.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die 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 die Werte in Klammern durch Informationen aus Ihrer Umgebung. Beispiel:

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

Erstellen Sie eine bedarfsgesteuerte Datensicherung

Sie können eine App jederzeit sichern.

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

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger laufenden s3-Backup-Vorgänge ausreichend ist. Wenn das Token während des Backup-Vorgangs abläuft, kann der Vorgang fehlschlagen.

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

Erstellen Sie eine Sicherung mithilfe eines CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-backup-cr.yaml`.
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.applicationRef**: (*Erforderlich*) Der Kubernetes-Name der zu sichernden Anwendung.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVault, wo die Sicherungsinhalte gespeichert werden sollen.
   - **spec.dataMover**: (*Optional*) Eine Zeichenkette, die angibt, welches Sicherungstool für den Sicherungsvorgang verwendet werden soll. Mögliche Werte (Groß-/Kleinschreibung):
     - `Restic`
     - `Kopia` (Standard)
   - **spec.reclaimPolicy**: (*Optional*) Definiert, was mit einem Backup geschieht, wenn es aus seinem Anspruch entfernt wird. Mögliche Werte:
     - `Delete`
     - `Retain` (Standard)
   - **spec.snapshotRef**: (*Optional*): Name des Snapshots, der als Quelle für das Backup verwendet werden soll. Wenn kein Name angegeben wird, wird ein temporärer Snapshot erstellt und gesichert.

     Beispiel YAML:

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

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

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

Erstellen Sie ein Backup mithilfe der CLI

Schritte

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

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

   Sie können optional das `--full-backup` Flag verwenden, um anzugeben, ob eine Sicherung nicht-inkrementell sein soll. Standardmäßig sind alle Backups inkrementell. Wenn dieses Flag verwendet wird, wird die Sicherung nicht-inkrementell. Es ist Best Practice, regelmäßig eine vollständige Sicherung durchzuführen und dazwischen inkrementelle Sicherungen zu erstellen, um das Risiko bei der Wiederherstellung zu minimieren.

Unterstützte Sicherungsanmerkungen

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

Anmerkung	Typ	Beschreibung	Standardwert
protect.trident.netapp.io/full-backup	Zeichenkette	Legt fest, ob eine Sicherung nicht inkrementell sein soll. Setzen Sie auf `true`, um eine nicht inkrementelle Sicherung zu erstellen. Es ist Best Practice, regelmäßig eine vollständige Sicherung durchzuführen und dazwischen inkrementelle Sicherungen zu erstellen, um das Risiko bei der Wiederherstellung zu minimieren.	"false"
protect.trident.netapp.io/snapshot-completion-timeout	Zeichenkette	Die maximal zulässige Zeit für den Abschluss des gesamten Snapshot-Vorgangs.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	Zeichenkette	Die maximal zulässige Zeitspanne, bis Volume-Snapshots den einsatzbereiten Zustand erreichen.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	Zeichenkette	Die maximal zulässige Zeit für die Erstellung von Volume-Snapshots.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	Zeichenkette	Maximale Zeit (in Sekunden), die auf neu erstellte PersistentVolumeClaims (PVCs) gewartet wird, um die `Bound` Phase zu erreichen, bevor der Vorgang fehlschlägt.	"1200" (20 Minuten)

Erstellen Sie einen Zeitplan für die Datensicherung

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

inkrementellen Backups dazwischen hilft, das Risiko bei Wiederherstellungen zu reduzieren.

- Sie können Zeitpläne für Snapshots nur erstellen, indem Sie `backupRetention` auf Null und `snapshotRetention` auf einen Wert größer als Null setzen. Das Setzen von `snapshotRetention` auf Null bedeutet, dass bei geplanten Backups zwar weiterhin Snapshots erstellt werden, diese jedoch temporär sind und unmittelbar nach Abschluss des Backups gelöscht werden.
- Clusterbezogene Ressourcen werden in eine Sicherung, einen Snapshot oder einen Klon aufgenommen, wenn sie in der Anwendungsdefinition explizit referenziert werden oder wenn sie Verweise auf einen der Anwendungs-Namespaces enthalten.

Erstellen Sie einen Zeitplan mithilfe eines CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-schedule-cr.yaml`.
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.dataMover**: (*Optional*) Eine Zeichenkette, die angibt, welches Sicherungstool für den Sicherungsvorgang verwendet werden soll. Mögliche Werte (Groß-/Kleinschreibung):
     - `Restic`
     - `Kopia` (Standard)
   - **spec.applicationRef**: Der Kubernetes-Name der zu sichernden Anwendung.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVault, wo die Sicherungsinhalte gespeichert werden sollen.
   - **spec.backupRetention**: (*Erforderlich*) Die Anzahl der aufzubewahrenden Backups. Null bedeutet, dass keine Backups erstellt werden sollen (nur Snapshots).
   - **spec.backupReclaimPolicy**: (*Optional*) Legt fest, was mit einem Backup geschieht, wenn die Backup-CR während der Aufbewahrungsfrist gelöscht wird. Nach Ablauf der Aufbewahrungsfrist werden Backups immer gelöscht. Mögliche Werte (Groß-/Kleinschreibung):
     - `Retain` (Standard)
     - `Delete`
   - **spec.snapshotRetention**: (*Erforderlich*) Die Anzahl der aufzubewahrenden Snapshots. Zero bedeutet, dass keine Snapshots erstellt werden sollen.
   - **spec.snapshotReclaimPolicy**: (*Optional*) Legt fest, was mit einem Snapshot geschieht, wenn die Snapshot-CR während seiner Aufbewahrungsfrist gelöscht wird. Nach Ablauf der Aufbewahrungsfrist werden Snapshots immer gelöscht. Mögliche Werte (Groß-/Kleinschreibung):
     - `Retain`
     - `Delete` (Standard)
   - **spec.granularity**: Die Häufigkeit, mit der der Zeitplan ausgeführt werden soll. Mögliche Werte sowie zugehörige Pflichtfelder:
     - `Hourly` (erfordert, dass Sie `spec.minute` angeben)
     - `Daily` (erfordert, dass Sie `spec.minute` und `spec.hour` angeben)
     - `Weekly` (erfordert, dass Sie `spec.minute, spec.hour` angeben, und `spec.dayOfWeek`)
     - `Monthly` (erfordert, dass Sie `spec.minute, spec.hour` angeben, und `spec.dayOfMonth`)
     - `Custom`
   - **spec.dayOfMonth**: (*Optional*) Der Tag des Monats (1 - 31), an dem der Zeitplan ausgeführt werden soll. Dieses Feld ist erforderlich, wenn die Granularität auf `Monthly` eingestellt ist. Der Wert muss als Zeichenkette angegeben werden.

- **spec.dayOfWeek**: (*Optional*) Der Wochentag (0 - 7), an dem der Zeitplan ausgeführt werden soll. Werte von 0 oder 7 bedeuten Sonntag. Dieses Feld ist erforderlich, wenn die Granularität auf `Weekly` eingestellt ist. Der Wert muss als Zeichenkette angegeben werden.
- **spec.hour**: (*Optional*) Die Stunde des Tages (0–23), zu der der Zeitplan ausgeführt werden soll. Dieses Feld ist erforderlich, wenn die Granularität auf `Daily`, `Weekly` oder `Monthly` eingestellt ist. Der Wert muss als Zeichenkette angegeben werden.
- **spec.minute**: (*Optional*) Die Minute der Stunde (0–59), zu der der Zeitplan ausgeführt werden soll. Dieses Feld ist erforderlich, wenn die Granularität auf `Hourly`, `Daily`, `Weekly` oder `Monthly` gesetzt ist. Der Wert muss als Zeichenkette angegeben werden.
- **spec.runImmediately**: (*Optional*) Auf `true` setzen, um beim Erstellen des Zeitplans einen einmaligen, sofortigen Baseline-Lauf (Backup und/oder Snapshot gemäß den Aufbewahrungseinstellungen) auszulösen. Standardmäßig `false`. Dies ändert die nachfolgende Wiederholung nicht.

  Beispiel-YAML für Backup- und Snapshot-Zeitplan:

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

  Beispiel-YAML für Snapshot-only-Zeitplan:

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

Beispiel-YAML für einen Zeitplan mit sofortiger Ausführung:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-daily-schedule-run-immediately
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "7"
  snapshotRetention: "7"
  granularity: Daily
  hour: "3"
  minute: "0"
  runImmediately: true
```

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

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

Erstellen Sie einen Zeitplan mithilfe der CLI

Schritte

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

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

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

Die folgenden Optionen bieten zusätzliche Kontrolle über Ihren Zeitplan:

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

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

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

Unterstützte Zeitplananmerkungen

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

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

Einen Snapshot löschen

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

Schritte

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

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

Ein Backup löschen

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

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

Schritte

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

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

Überprüfen Sie den Status eines Sicherungsvorgangs

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

Schritte

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

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

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

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

Bevor Sie beginnen

Stellen Sie Folgendes sicher:

- Sie haben Trident Protect installiert.
- Sie haben eine Anwendung in Trident Protect definiert. Diese Anwendung verfügt nur über eingeschränkte Schutzfunktionen, bis Sie dieses Verfahren abgeschlossen haben.
- Sie haben `azure-netapp-files` als Standard-Speicherklasse für Ihr Speicher-Backend ausgewählt.

Für Konfigurationsschritte erweitern

1. Führen Sie Folgendes in Trident aus, wenn das ANF-Volume vor dem Upgrade auf Trident 24.10 erstellt wurde:

   a. Aktivieren Sie das Snapshot-Verzeichnis für jedes PV, das auf azure-netapp-files basiert und der Anwendung zugeordnet ist:

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

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

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

   Antwort:

   ```
   snapshotDirectory: "true"
   ```

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

Ergebnis

Die Anwendung ist für Backup und Restore mit Trident Protect bereit. Jede PVC kann auch von anderen Anwendungen für Backups und Restores verwendet werden.

Anwendungen wiederherstellen

Anwendungen mit Trident Protect wiederherstellen

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

- Wenn Sie eine Anwendung wiederherstellen, werden alle für die Anwendung konfigurierten Ausführungs-Hooks mit der Anwendung wiederhergestellt. Wenn ein Ausführungs-Hook nach der Wiederherstellung vorhanden ist, wird er automatisch als Teil des Wiederherstellungsvorgangs ausgeführt.
- Die Wiederherstellung aus einem Backup in einen anderen Namespace oder in den ursprünglichen Namespace wird für qtree Volumes unterstützt. Die Wiederherstellung aus einem Snapshot in einen anderen Namespace oder in den ursprünglichen Namespace wird für qtree Volumes jedoch nicht unterstützt.
- Sie können die Wiederherstellungsvorgänge mithilfe erweiterter Einstellungen anpassen. Weitere Informationen finden Sie unter "Verwenden Sie die erweiterten Trident Protect-Wiederherstellungseinstellungen".

Wiederherstellung aus einem Backup in einen anderen Namensraum

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

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

Bevor Sie beginnen

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

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

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

Verwenden Sie einen CR

Schritte

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

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

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

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

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

   Trident Protect wählt bestimmte Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn Sie beispielsweise eine Ressource für einen persistenten Volume-Claim auswählen und diese einen zugehörigen Pod hat, wird Trident Protect auch den zugehörigen Pod wiederherstellen.

- **resourceFilter.resourceSelectionCriteria**: (Für die Filterung erforderlich) Verwenden Sie `Include` oder `Exclude`, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:
  - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese mit einer ODER-Verknüpfung verglichen und die Felder innerhalb jedes Elements (group, kind, version) werden mit einer UND-Verknüpfung verglichen.
    - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressource.
    - **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
    - **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
    - **resourceMatchers[].names**: (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
    - **resourceMatchers[].namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
    - **resourceMatchers[].labelSelectors**: (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

      Beispiel:

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

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

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

Verwenden Sie die Befehlszeile

Schritte

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

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

Stellen Sie aus einer Sicherung in den ursprünglichen Namensraum wieder her

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

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

Bevor Sie beginnen

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

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

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

Verwenden Sie einen CR

Schritte

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

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

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

     Beispiel:

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

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

   Trident Protect wählt bestimmte Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn Sie beispielsweise eine Ressource für einen persistenten Volume-Claim auswählen und diese einen zugehörigen Pod hat, wird Trident Protect auch den zugehörigen Pod wiederherstellen.

   - **resourceFilter.resourceSelectionCriteria**: (Für die Filterung erforderlich) Verwenden Sie `Include` oder `Exclude`, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:
     - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese mit einer ODER-Verknüpfung verglichen und die Felder innerhalb jedes Elements (group, kind, version) werden mit einer UND-Verknüpfung verglichen.

- **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressource.
- **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
- **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
- **resourceMatchers[].names**: (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].labelSelectors**: (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

  Beispiel:

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

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

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

Verwenden Sie die Befehlszeile

Schritte

1. Stellen Sie die Sicherung im ursprünglichen Namensraum wieder her, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Das `backup` Argument verwendet einen Namensraum und einen Sicherungsnamen im Format `<namespace>/<name>`. Beispiel:

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

Wiederherstellung aus einem Backup auf einem anderen Cluster

Sie können ein Backup auf einem anderen Cluster wiederherstellen, wenn es ein Problem mit dem ursprünglichen Cluster gibt.

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

Bevor Sie beginnen

Stellen Sie sicher, dass die folgenden Voraussetzungen erfüllt sind:

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

Schritte

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

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

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

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

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

Durch Ausführen dieses Befehls werden die verfügbaren Backups im AppVault angezeigt, einschließlich ihrer Ursprungscluster, entsprechenden Anwendungsnamen, Zeitstempel und Archivpfade.

Beispielausgabe:

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

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

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

Verwenden Sie einen CR

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-backup-restore-cr.yaml`.
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVault, in dem die Sicherungsinhalte gespeichert sind.
   - **spec.appArchivePath**: (*Erforderlich*) Der Pfad innerhalb von AppVault, in dem die Sicherungsinhalte gespeichert sind. Verwenden Sie den Befehl aus Schritt 2, um die Sicherungsinhalte anzuzeigen und `appArchivePath`für die Sicherung zu finden, die Sie wiederherstellen möchten.
   - **spec.destinationApplicationName**: (*Optional*) Der Name für die wiederhergestellte Anwendung. Falls angegeben, verwendet die wiederhergestellte Anwendung diesen Namen. Falls nicht angegeben, verwendet die wiederhergestellte Anwendung den Namen der Quellanwendung.
   - **spec.namespaceMapping**: Die Zuordnung des Quell-Namespace des Wiederherstellungsvorgangs zum Ziel-Namespace. Ersetzen Sie `my-source-namespace` und `my-destination-namespace` durch Informationen aus Ihrer Umgebung.

     Beispiel:

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

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

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

Verwenden Sie die Befehlszeile

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

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

Wiederherstellung aus einem Snapshot in einen anderen Namespace

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

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

Bevor Sie beginnen

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

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

Verwenden Sie einen CR

Schritte

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

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

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

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

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

   > Trident Protect wählt bestimmte Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn Sie beispielsweise eine Ressource für einen persistenten Volume-Claim auswählen und diese einen zugehörigen Pod hat, wird Trident Protect auch den zugehörigen Pod wiederherstellen.

   - **resourceFilter.resourceSelectionCriteria**: (Für die Filterung erforderlich) Verwenden Sie

`Include` oder `Exclude`, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:

- **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese mit einer ODER-Verknüpfung verglichen und die Felder innerhalb jedes Elements (group, kind, version) werden mit einer UND-Verknüpfung verglichen.
  - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressource.
  - **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
  - **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
  - **resourceMatchers[].names**: (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
  - **resourceMatchers[].namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
  - **resourceMatchers[].labelSelectors**: (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

    Beispiel:

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

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

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

Verwenden Sie die Befehlszeile

Schritte

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

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

     Beispiel:

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

Wiederherstellung aus einem Snapshot in den ursprünglichen Namensraum

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

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

Bevor Sie beginnen

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

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

Verwenden Sie einen CR

Schritte

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

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

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

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

   Trident Protect wählt bestimmte Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn Sie beispielsweise eine Ressource für einen persistenten Volume-Claim auswählen und diese einen zugehörigen Pod hat, wird Trident Protect auch den zugehörigen Pod wiederherstellen.

   - **resourceFilter.resourceSelectionCriteria**: (Für die Filterung erforderlich) Verwenden Sie `Include` oder `Exclude`, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:
     - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese mit einer ODER-Verknüpfung verglichen und die Felder innerhalb jedes Elements (group, kind, version) werden mit einer UND-Verknüpfung verglichen.
       - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressource.
       - **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.

- **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
- **resourceMatchers[].names**: (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].labelSelectors**: (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

  Beispiel:

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

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

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

Verwenden Sie die Befehlszeile

Schritte

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

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

Überprüfen Sie den Status eines Wiederherstellungsvorgangs

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

Schritte

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

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

Verwenden Sie die erweiterten Trident Protect-Wiederherstellungseinstellungen

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

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

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

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

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

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

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

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

Beispiel

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

Vor dem Wiederherstellungs- oder Failover-Vorgang

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

Namensraum	Anmerkungen	Etiketten
Namespace ns-1 (Quelle)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• environment=production • compliance=hipaa • name=ns-1
Namespace ns-2 (Ziel)	• annotation.one/key: "true" • annotation.three/key: "false"	• role=database

Nach dem Wiederherstellungsvorgang

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

Namensraum	Anmerkungen	Etiketten
Namespace ns-2 (Ziel)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • compliance=hipaa • environment=production • role=database

Unterstützte Felder

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

Speicherklassenzuordnung

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

Beispiel:

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

Unterstützte Annotationen

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

Anmerkung	Typ	Beschreibung	Standardwert
protect.trident.netapp.io/data-mover-timeout-sec	Zeichenkette	Die maximal zulässige Zeit (in Sekunden), in der der Datenübertragungsvorgang angehalten werden darf.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	Zeichenkette	Die maximale Größenbeschränkung (in Megabytes) für den Kopia-Inhaltscache.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	Zeichenkette	Maximale Zeit (in Sekunden), die auf neu erstellte PersistentVolumeClaims (PVCs) gewartet wird, um die `Bound` Phase zu erreichen, bevor der Vorgang fehlschlägt. Gilt für alle Restore-CR-Typen (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Verwenden Sie einen höheren Wert, wenn Ihr Storage-Backend oder Cluster häufig mehr Zeit benötigt.	"1200" (20 Minuten)

Anwendungen mit NetApp SnapMirror und Trident Protect replizieren

Mit Trident Protect können Sie die asynchronen Replizierungsfunktionen der NetApp SnapMirror-Technologie nutzen, um Daten und Anwendungsänderungen von einem

Storage-Backend auf ein anderes zu replizieren, entweder im selben Cluster oder zwischen verschiedenen Clustern.

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

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

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

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

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

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

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

Beispiel

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

Vor dem Wiederherstellungs- oder Failover-Vorgang

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

Namensraum	Anmerkungen	Etiketten
Namespace ns-1 (Quelle)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• environment=production • compliance=hipaa • name=ns-1
Namespace ns-2 (Ziel)	• annotation.one/key: "true" • annotation.three/key: "false"	• role=database

Nach dem Wiederherstellungsvorgang

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

Namensraum	Anmerkungen	Etiketten
Namespace ns-2 (Ziel)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • compliance=hipaa • environment=production • role=database

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

Ausführungshooks während Failover- und Reverse-Operationen

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

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

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

Eine Replikationsbeziehung einrichten

Die Einrichtung einer Replikationsbeziehung umfasst Folgendes:

- Auswahl, wie häufig Trident Protect einen App-Snapshot erstellen soll (der sowohl die Kubernetes-Ressourcen der App als auch die Volume-Snapshots für jedes der App-Volumes umfasst)
- Auswahl des Replikationszeitplans (einschließlich Kubernetes-Ressourcen sowie persistenter Volume-Daten)
- Einstellen des Zeitpunkts für die Aufnahme des Snapshots

Schritte

1. Erstellen Sie auf dem Quell-Cluster ein AppVault für die Quellanwendung. Passen Sie je nach Speicheranbieter ein Beispiel in "AppVault benutzerdefinierte Ressourcen" an Ihre Umgebung an:

Erstellen Sie ein AppVault mit einem CR

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

b. Konfigurieren Sie die folgenden Attribute:

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

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

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

Erstellen Sie ein AppVault mit der CLI

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

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

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

Erstellen Sie die Quellanwendung mithilfe eines CR

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

b. Konfigurieren Sie die folgenden Attribute:

- **metadata.name**: (*Erforderlich*) Der Name der benutzerdefinierten Anwendungsressource. Notieren Sie sich den Namen, den Sie wählen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diesen Wert verweisen.
- **spec.includedNamespaces**: (*Erforderlich*) Ein Array von Namespaces und zugehörigen Labels. Verwenden Sie Namespace-Namen und grenzen Sie optional den Geltungsbereich der Namespaces mit Labels ein, um Ressourcen anzugeben, die in den hier aufgeführten Namespaces vorhanden sind. Der Anwendungsnamespace muss Teil dieses Arrays sein.

  Beispiel YAML:

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

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

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

Erstellen Sie die Quellanwendung mithilfe der CLI

a. Erstellen Sie die Quellanwendung. Beispiel:

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

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

4. Erstellen Sie auf dem Quell-Cluster den Replikationszeitplan CR:

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

Im Falle eines Failovers kann das System diese Snapshots bis zu 7 Tage lang für Rückoperationen nutzen. Dieser Ansatz macht den Rückprozess schneller und effizienter, da nur die seit dem letzten Snapshot vorgenommenen Änderungen übertragen werden, nicht alle Daten.

Wenn ein bestehender Zeitplan für die Anwendung bereits die gewünschten Aufbewahrungsanforderungen erfüllt, sind keine zusätzlichen Zeitpläne erforderlich.

Erstellen Sie den Replikationszeitplan mithilfe einer CR

a. Erstellen Sie einen Replikationszeitplan für die Quellanwendung:

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

ii. Konfigurieren Sie die folgenden Attribute:

- **metadata.name**: (*Erforderlich*) Der Name der benutzerdefinierten Zeitplanressource.
- **spec.appVaultRef**: (*Erforderlich*) Dieser Wert muss mit dem Feld metadata.name der AppVault für die Quellanwendung übereinstimmen.
- **spec.applicationRef**: (*Erforderlich*) Dieser Wert muss mit dem Feld metadata.name der Quellanwendung CR übereinstimmen.
- **spec.backupRetention**: (*Erforderlich*) Dieses Feld ist erforderlich und der Wert muss auf 0 gesetzt werden.
- **spec.enabled**: Muss auf true gesetzt werden.
- **spec.granularity**: Muss auf `Custom` gesetzt werden.
- **spec.recurrenceRule**: Definieren Sie ein Startdatum in UTC-Zeit und ein Wiederholungsintervall.
- **spec.snapshotRetention**: Muss auf 2 eingestellt sein.

Beispiel YAML:

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

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

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

Erstellen Sie den Replikationszeitplan mithilfe der CLI

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

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

Beispiel:

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

5. Erstellen Sie auf dem Ziel-Cluster eine Quellanwendungs-AppVault-CR, die mit der AppVault-CR identisch ist, die Sie auf dem Quell-Cluster angewendet haben, und benennen Sie sie (zum Beispiel `trident-protect-appvault-primary-destination.yaml`).
6. Wenden Sie die CR an:

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

7. Erstellen Sie eine Ziel-AppVault CR für die Zielanwendung im Ziel-Cluster. Passen Sie je nach Speicheranbieter ein Beispiel in "AppVault benutzerdefinierte Ressourcen" an Ihre Umgebung an:
   a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (zum Beispiel `trident-protect-appvault-secondary-destination.yaml`).
   b. Konfigurieren Sie die folgenden Attribute:
      - **metadata.name**: (*Erforderlich*) Der Name der AppVault Custom Resource. Notieren Sie sich den Namen, den Sie wählen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diesen Wert verweisen.
      - **spec.providerConfig**: (*Erforderlich*) Speichert die Konfiguration, die für den Zugriff auf AppVault mit dem angegebenen Provider erforderlich ist. Wählen Sie einen `bucketName` und alle weiteren erforderlichen Details für Ihren Provider. Notieren Sie sich die Werte, die Sie wählen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diese Werte verweisen. Siehe "AppVault benutzerdefinierte Ressourcen" für Beispiele von AppVault CRs mit anderen Providern.

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

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

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

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

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

Erstellen Sie eine AppMirrorRelationship mit einem CR

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

b. Konfigurieren Sie die folgenden Attribute:

- **metadata.name:** (Erforderlich) Der Name der AppMirrorRelationship benutzerdefinierten Ressource.
- **spec.destinationAppVaultRef**: (*Erforderlich*) Dieser Wert muss mit dem Namen der AppVault für die Zielanwendung auf dem Ziel-Cluster übereinstimmen.
- **spec.namespaceMapping**: (*Erforderlich*) Die Ziel- und Quell-Namespaces müssen mit dem in der jeweiligen Anwendungs-CR definierten Anwendungs-Namespace übereinstimmen.
- **spec.sourceAppVaultRef**: (*Erforderlich*) Dieser Wert muss mit dem Namen der AppVault für die Quellanwendung übereinstimmen.
- **spec.sourceApplicationName**: (*Erforderlich*) Dieser Wert muss mit dem Namen der Quellanwendung übereinstimmen, die Sie im Quellanwendungs-CR definiert haben.
- **spec.sourceApplicationUID**: (Erforderlich) Dieser Wert muss mit der UID der Quellanwendung übereinstimmen, die Sie im Quellanwendungs-CR definiert haben.
- **spec.storageClassName**: (*Optional*) Wählen Sie den Namen einer gültigen Speicherklasse auf dem Cluster. Die Speicherklasse muss mit einer ONTAP Storage-VM verknüpft sein, die mit der Quellumgebung gepaart ist. Wenn keine Speicherklasse angegeben wird, wird standardmäßig die Standard-Speicherklasse auf dem Cluster verwendet.
- **spec.recurrenceRule**: Definieren Sie ein Startdatum in UTC-Zeit und ein Wiederholungsintervall.

  Beispiel YAML:

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

c. Nachdem Sie die `trident-protect-relationship.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die 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, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen:

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

Beispiel:

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

9. (*Optional*) Überprüfen Sie auf dem Ziel-Cluster den Status und den Zustand der Replikationsbeziehung:

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

Failover zum Ziel-Cluster

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

Schritte

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

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

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

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

Eine fehlgeschlagene Replikationsbeziehung erneut synchronisieren

Der Resynchronisierungsvorgang stellt die Replikationsbeziehung wieder her. Nach der Durchführung eines Resynchronisierungsvorgangs wird die ursprüngliche Quellanwendung zur aktiven Anwendung und alle Änderungen, die an der aktiven Anwendung im Ziel-Cluster vorgenommen wurden, werden verworfen.

Der Prozess stoppt die App auf dem Ziel-Cluster, bevor die Replikation wiederhergestellt wird.

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

Schritte

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

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

5. Wenn Sie auf dem Ziel-Cluster Schutzzeitpläne für die ausgefallene Anwendung erstellt haben, entfernen Sie diese. Alle verbleibenden Zeitpläne verursachen Fehler bei Volume-Snapshots.

Reverse resync einer fehlgeschlagenen Replikationsbeziehung

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

Schritte

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

Replikationsrichtung der Anwendung umkehren

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

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

Schritte

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

Erstellen Sie einen Shutdown-Snapshot mithilfe eines CR

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

b. Erstellen Sie eine ShutdownSnapshot CR-Datei:

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

   ii. Konfigurieren Sie die folgenden Attribute:

   - **metadata.name**: (*Erforderlich*) Der Name der benutzerdefinierten Ressource.
   - **spec.AppVaultRef**: (*Erforderlich*) Dieser Wert muss mit dem Feld metadata.name der AppVault für die Quellanwendung übereinstimmen.
   - **spec.ApplicationRef**: (*Erforderlich*) Dieser Wert muss mit dem Feld metadata.name der Quellanwendungs-CR-Datei übereinstimmen.

   Beispiel YAML:

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

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

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

Erstellen Sie einen Shutdown-Snapshot mit der CLI

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

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

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

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

3. Ermitteln Sie auf dem Quell-Cluster den Wert von **shutdownsnapshot.status.appArchivePath** mit dem folgenden Befehl und notieren Sie den letzten Teil des Dateipfads (auch Basisname genannt; dies ist alles nach dem letzten Schrägstrich):

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

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

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

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

Ergebnis

Die folgenden Aktionen erfolgen aufgrund der umgekehrten Replikation:

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

Failback von Applikationen auf den ursprünglichen Quell-Cluster

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

Dieser Prozess beginnt mit einer Beziehung, die einen Failover auf ein Ziel abgeschlossen hat und umfasst die folgenden Schritte:

- Beginnen Sie mit einem übergewechselten Zustand.
- Reverse resync die Replikationsbeziehung.

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

- Die Replikationsrichtung umkehren.

Schritte

1. Führen Sie die Reverse resync einer fehlgeschlagenen Replikationsbeziehung Schritte aus.
2. Führen Sie die Replikationsrichtung der Anwendung umkehren Schritte aus.

Eine Replikationsbeziehung löschen

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

Schritte

1. Löschen Sie im aktuellen Ziel-Cluster die AppMirrorRelationship CR:

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

Anwendungen mit Trident Protect migrieren

Sie können Ihre Anwendungen zwischen Clustern oder auf verschiedene Storage-Klassen migrieren, indem Sie Sicherungsdaten wiederherstellen.

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

Sicherungs- und Wiederherstellungsvorgänge

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

Klon auf denselben Cluster

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

Schritte

1. Führen Sie einen der folgenden Schritte aus:
   a. "Erstellen Sie einen Snapshot".
   b. "Erstelle eine Sicherungskopie".
2. Führen Sie auf demselben Cluster einen der folgenden Schritte aus, je nachdem, ob Sie einen Snapshot oder ein Backup erstellt haben:

a. "Stellen Sie Ihre Daten aus dem Snapshot wieder her".

b. "Stellen Sie Ihre Daten aus der Sicherung wieder her".

Klonen auf anderen Cluster

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

Sie können eine Anwendung zwischen verschiedenen Clustern replizieren, indem Sie "SnapMirror-Replikation".

Schritte

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

Anwendungen von einer Speicherklasse zu einer anderen Speicherklasse migrieren

Sie können Anwendungen von einer Speicherklasse in eine andere Speicherklasse migrieren, indem Sie ein Backup in der Ziel-Speicherklasse wiederherstellen.

Zum Beispiel (ohne die Geheimnisse aus dem Wiederherstellungs-CR):

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

Stellen Sie den Snapshot mithilfe einer CR wieder her

Schritte

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

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

   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVault, in dem die Snapshot-Inhalte gespeichert sind.
   - **spec.namespaceMapping**: Die Zuordnung des Quell-Namespace des Wiederherstellungsvorgangs zum Ziel-Namespace. Ersetzen Sie `my-source-namespace` und `my-destination-namespace` durch Informationen aus Ihrer Umgebung.

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

3. Optional können Sie, falls Sie nur bestimmte Ressourcen der Anwendung für die Wiederherstellung auswählen müssen, Filter hinzufügen, die Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließen:
   - **resourceFilter.resourceSelectionCriteria**: (Für die Filterung erforderlich) Verwenden Sie `include or exclude` , um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:
     - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese mit einer ODER-Verknüpfung verglichen und die Felder innerhalb jedes Elements (group, kind, version) werden mit einer UND-Verknüpfung verglichen.
       - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressource.

- **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
- **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
- **resourceMatchers[].names**: (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].labelSelectors**: (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

  Beispiel:

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

4. Nachdem Sie die `trident-protect-snapshot-restore-cr.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die 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, wobei Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen.

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

Beispiel:

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

Trident Protect-Ausführungshooks verwalten

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

Arten von Execution Hooks

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

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

Ausführungsreihenfolge

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

1. Alle anwendbaren benutzerdefinierten Pre-Operation-Hooks werden in den entsprechenden Containern ausgeführt. Sie können beliebig viele benutzerdefinierte Pre-Operation-Hooks erstellen und ausführen, aber die Ausführungsreihenfolge dieser Hooks vor der Operation ist weder garantiert noch konfigurierbar.
2. Gegebenenfalls kommt es zu Dateisystem-Freezes. "Erfahren Sie mehr über die Konfiguration des Einfrierens von Dateisystemen mit Trident Protect".
3. Der Datenschutzvorgang wird durchgeführt.
4. Eingefrorene Dateisysteme werden, falls zutreffend, wieder freigegeben.
5. Alle anwendbaren benutzerdefinierten Nachbearbeitungs-Hooks werden in den entsprechenden Containern ausgeführt. Sie können beliebig viele benutzerdefinierte Nachbearbeitungs-Hooks erstellen und ausführen, aber die Ausführungsreihenfolge dieser Hooks nach der Operation ist weder garantiert noch konfigurierbar.

Wenn Sie mehrere Ausführungs-Hooks desselben Typs erstellen (z. B. pre-snapshot), ist die

Ausführungsreihenfolge dieser Hooks nicht garantiert. Die Ausführungsreihenfolge von Hooks unterschiedlicher Typen ist jedoch garantiert. Zum Beispiel ist dies die Ausführungsreihenfolge einer Konfiguration, die alle verschiedenen Hook-Typen enthält:

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

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

Sie sollten Ihre Ausführungshook-Skripte immer testen, bevor Sie sie in einer Produktionsumgebung aktivieren. Sie können den Befehl 'kubectl exec' verwenden, um die Skripte bequem zu testen. Nachdem Sie die Ausführungshooks in einer Produktionsumgebung aktiviert haben, testen Sie die resultierenden Snapshots und Backups, um sicherzustellen, dass sie konsistent sind. Sie können dies tun, indem Sie die App in einen temporären Namespace klonen, den Snapshot oder das Backup wiederherstellen und dann die App testen.

Wenn ein Pre-Snapshot-Ausführungs-Hook Kubernetes-Ressourcen hinzufügt, ändert oder entfernt, werden diese Änderungen in den Snapshot oder das Backup und in jede nachfolgende Wiederherstellungsoperation einbezogen.

Wichtige Hinweise zu benutzerdefinierten Ausführungshooks

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

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

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

Ausführungs-Hook-Filter

Wenn Sie einen Ausführungshook für eine Anwendung hinzufügen oder bearbeiten, können Sie Filter zum Ausführungshook hinzufügen, um zu steuern, auf welche Container der Hook angewendet wird. Filter sind nützlich für Anwendungen, die auf allen Containern dasselbe Container-Image verwenden, aber jedes Image für einen anderen Zweck nutzen (wie zum Beispiel Elasticsearch). Filter ermöglichen es Ihnen, Szenarien zu

erstellen, in denen Ausführungshooks auf einigen, aber nicht unbedingt allen identischen Containern ausgeführt werden. Wenn Sie mehrere Filter für einen einzelnen Ausführungshook erstellen, werden diese mit einem logischen UND-Operator kombiniert. Sie können bis zu 10 aktive Filter pro Ausführungshook haben.

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

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

Beispiele für Execution Hooks

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

Erstellen Sie einen Ausführungs-Hook

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

Verwenden Sie einen CR

Schritte

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

   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.applicationRef**: (*Erforderlich*) Der Kubernetes-Name der Anwendung, für die der Ausführungshook ausgeführt werden soll.
   - **spec.stage**: (*Erforderlich*) Eine Zeichenkette, die angibt, in welcher Phase der Aktion der Ausführungs-Hook ausgeführt werden soll. Mögliche Werte:
     - Vor
     - Beitrag
   - **spec.action**: (*Erforderlich*) Eine Zeichenkette, die angibt, welche Aktion der Ausführungs-Hook ausführt, vorausgesetzt, dass alle angegebenen Ausführungs-Hook-Filter übereinstimmen. Mögliche Werte:
     - Schnappschuss
     - Backup
     - Wiederherstellen
     - Failover
   - **spec.enabled**: (*Optional*) Gibt an, ob dieser Ausführungs-Hook aktiviert oder deaktiviert ist. Wenn nicht angegeben, ist der Standardwert true.
   - **spec.hookSource**: (*Erforderlich*) Eine Zeichenkette, die das Base64-kodierte Hook-Skript enthält.
   - **spec.timeout**: (*Optional*) Eine Zahl, die angibt, wie lange in Minuten der Ausführungs-Hook ausgeführt werden darf. Der Mindestwert beträgt 1 Minute und der Standardwert beträgt 25 Minuten, falls nicht angegeben.
   - **spec.arguments**: (*Optional*) Eine YAML-Liste von Argumenten, die Sie für den Ausführungshook angeben können.
   - **spec.matchingCriteria**: (*Optional*) Eine optionale Liste von Kriterien-Schlüssel-Wert-Paaren, wobei jedes Paar einen Ausführungs-Hook-Filter bildet. Sie können bis zu 10 Filter pro Ausführungs-Hook hinzufügen.
   - **spec.matchingCriteria.type**: (*Optional*) Eine Zeichenkette, die den Filtertyp des Ausführungshooks identifiziert. Mögliche Werte:
     - ContainerImage
     - ContainerName
     - PodName
     - PodLabel
     - NamespaceName
   - **spec.matchingCriteria.value**: (*Optional*) Eine Zeichenkette oder ein regulärer Ausdruck, der den Wert des Ausführungs-Hook-Filters identifiziert.

Beispiel YAML:

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

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

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

Verwenden Sie die Befehlszeile

Schritte

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

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

Einen Ausführungshook manuell ausführen

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

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

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

Schritt 1: Erstellen Sie eine Ressourcensicherung

Verwenden Sie einen CR

Schritte

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

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

     Beispiel YAML:

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

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

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

Verwenden Sie die Befehlszeile

Schritte

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

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

2. Zeigen Sie den Status der Datensicherung an. Sie können diesen Beispielbefehl wiederholt verwenden, bis der Vorgang abgeschlossen ist:

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

3. Überprüfen Sie, ob das Backup erfolgreich war:

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

Schritt 2: Den Ausführungshook ausführen

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-hook-run.yaml`.
2. Konfigurieren Sie die folgenden Attribute so, dass sie Ihrer Trident Protect-Umgebung und Clusterkonfiguration entsprechen:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.applicationRef**: (*Erforderlich*) Stellen Sie sicher, dass dieser Wert mit dem Anwendungsnamen aus dem ResourceBackup CR übereinstimmt, den Sie in Schritt 1 erstellt haben.
   - **spec.appVaultRef**: (*Erforderlich*) Stellen Sie sicher, dass dieser Wert mit dem appVaultRef aus dem ResourceBackup CR übereinstimmt, den Sie in Schritt 1 erstellt haben.
   - **spec.appArchivePath**: Stellen Sie sicher, dass dieser Wert mit dem appArchivePath aus dem ResourceBackup CR übereinstimmt, den Sie in Schritt 1 erstellt haben.

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

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

     Beispiel YAML:

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

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

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

Verwenden Sie die Befehlszeile

Schritte

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

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

2. Überprüfen Sie den Status des ausgeführten Hooks. Sie können diesen Befehl so lange wiederholen, bis der Vorgang abgeschlossen ist:

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

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

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

Trident Protect deinstallieren

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

Um Trident Protect zu entfernen, führen Sie die folgenden Schritte aus.

Schritte

1. Entfernen Sie die Trident Protect CR-Dateien:

   Dieser Schritt ist für Version 25.06 und höher nicht erforderlich.

   ```
   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
   ```