Nutzen Sie Astra Trident

Astra Trident

NetApp
November 20, 2023

Inhalt

Nutzen Sie Astra Trident

Back-Ends konfigurieren

Ein Backend definiert die Beziehung zwischen Astra Trident und einem Storage-System. Er erzählt Astra Trident, wie man mit diesem Storage-System kommuniziert und wie Astra Trident Volumes darauf bereitstellen sollte. Astra Trident bietet automatisch Storage-Pools aus Back-Ends an, die den von einer Storage-Klasse definierten Anforderungen entsprechen. Erfahren Sie mehr über die Konfiguration des Backend auf der Grundlage des jeweiligen Storage-Systems.

- "Konfigurieren Sie ein Azure NetApp Files-Backend"
- "Konfigurieren Sie ein Cloud Volumes Service für AWS-Backend"
- "Konfigurieren Sie ein Back-End für Cloud Volumes Service für Google Cloud Platform"
- "Konfigurieren Sie ein NetApp HCI- oder SolidFire-Backend"
- "Konfigurieren Sie ein Backend mit ONTAP-NAS-Treibern"
- "Konfigurieren Sie ein Backend mit ONTAP-SAN-Treibern"
- "Setzen Sie Astra Trident mit Amazon FSX für NetApp ONTAP ein"

Konfigurieren Sie ein Azure NetApp Files-Backend

Erfahren Sie, wie Sie Azure NetApp Files (ANF) mit den angegebenen Beispielkonfigurationen als Backend für Ihre Astra Trident Installation konfigurieren.

> Der Azure NetApp Files-Service unterstützt keine Volumes mit weniger als 100 GB. Astra Trident erstellt automatisch 100-GB-Volumes, wenn ein kleineres Volume benötigt wird.

Was Sie benötigen

Um ein zu konfigurieren und zu verwenden "Azure NetApp Dateien" Back-End, Sie benötigen Folgendes:

- `subscriptionID` Über ein Azure Abonnement mit aktiviertem Azure NetApp Files.
- `tenantID`, `clientID`, und `clientSecret` Von einem "App-Registrierung" In Azure Active Directory mit ausreichenden Berechtigungen für den Azure NetApp Files-Service. Die App-Registrierung sollte das verwenden `Owner` Oder `Contributor` Rolle, die von Azure vordefiniert ist.

  > Weitere Informationen zu den integrierten Azure-Rollen finden Sie im "Azure-Dokumentation".

- Im Azure `location` Das enthält mindestens eine "Delegiertes Subnetz".
- Wenn Sie Azure NetApp Files zum ersten Mal oder an einem neuen Standort verwenden, ist eine Erstkonfiguration erforderlich. Siehe "quickstart-Anleitung".

Über diese Aufgabe

Basierend auf der Back-End-Konfiguration (Subnetz, virtuelles Netzwerk, Service Level und Standort) erstellt Trident ANF Volumes auf Kapazitäts-Pools, die am angeforderten Standort verfügbar sind und dem angeforderten Service Level und Subnetz entsprechen.

Astra Trident 21.04.0 und frühere Versionen unterstützen keine manuellen QoS-Kapazitäts-Pools.

Back-End-Konfigurationsoptionen

Die Back-End-Konfigurationsoptionen finden Sie in der folgenden Tabelle:

Parameter	Beschreibung	Standard
`version`		Immer 1
`storageDriverName`	Name des Speichertreibers	„azure-netapp-Files“
`backendName`	Benutzerdefinierter Name oder das Storage-Backend	Treibername + „_“ + zufällige Zeichen
`subscriptionID`	Die Abonnement-ID Ihres Azure Abonnements	
`tenantID`	Die Mandanten-ID aus einer App-Registrierung	
`clientID`	Die Client-ID aus einer App-Registrierung	
`clientSecret`	Der Client-Schlüssel aus einer App-Registrierung	
`serviceLevel`	Einer von `Standard`, `Premium`, Oder `Ultra`	„“ (zufällig)
`location`	Name des Azure Speicherorts, an dem die neuen Volumes erstellt werden	„“ (zufällig)
`virtualNetwork`	Name eines virtuellen Netzwerks mit einem delegierten Subnetz	„“ (zufällig)
`subnet`	Name eines an delegierten Subnetzes `Microsoft.Netapp/volumes`	„“ (zufällig)
`nfsMountOptions`	Engmaschige Kontrolle der NFS-Mount-Optionen	„Nfsvers=3“
`limitVolumeSize`	Bereitstellung fehlgeschlagen, wenn die angeforderte Volume-Größe über diesem Wert liegt	„“ (nicht standardmäßig durchgesetzt)
`debugTraceFlags`	Fehler-Flags bei der Fehlerbehebung beheben. Beispiel: `\{"api":false, "method":true}`. Verwenden Sie dies nur, wenn Sie Fehler beheben und einen detaillierten Log Dump benötigen.	Null

Ändern des `capacityPools` Feld in einem vorhandenen Back-End, d. h. die Verringerung der Anzahl der für die Bereitstellung verwendeten Kapazitäts-Pools, führt zu verwaisten Volumes, die im Kapazitäts-Pool/-Pools bereitgestellt werden, die nicht Teil des sind `capacityPools` Liste hinzufügen. Das Klonen dieser verwaisten Volumes schlägt fehl.

Wenn beim Versuch, ein PVC zu erstellen, ein Fehler „Keine Kapazitätspools gefunden“ auftritt, ist es wahrscheinlich, dass Ihre App-Registrierung nicht über die erforderlichen Berechtigungen und Ressourcen (Subnetz, virtuelles Netzwerk, Kapazitäts-Pool) verbunden ist. Astra Trident protokolliert die Azure Ressourcen, die es entdeckt hat, wenn das Backend erstellt wird, wenn Debug aktiviert ist. Prüfen Sie, ob eine geeignete Rolle verwendet wird.

Wenn Sie Volumes mit NFS Version 4.1 mounten möchten, können Sie die Volumes mit einbeziehen `nfsvers=4` Wählen Sie in der Liste mit durch Komma getrennten Mount-Optionen NFS v4.1 aus. Alle in einer Speicherklasse festgelegten Mount-Optionen überschreiben die Mount-Optionen, die in einer Back-End-Konfigurationsdatei festgelegt sind.

Sie können festlegen, wie jedes Volume standardmäßig bereitgestellt wird, indem Sie die folgenden Optionen in einem speziellen Abschnitt der Konfigurationsdatei angeben. Sehen Sie sich die Konfigurationsbeispiele unten an.

Parameter	Beschreibung	Standard
`exportRule`	Die Exportregel(n) für neue Volumes	„0.0.0.0/0“
`size`	Die Standardgröße der neuen Volumes	„100 GB“

Der `exportRule` Wert muss eine kommagetrennte Liste beliebiger Kombinationen von IPv4-Adressen oder IPv4-Subnetzen in CIDR-Notation sein.

Astra Trident kopiert bei allen auf einem ANF-Backend erstellten Volumes alle auf einem Storage-Pool vorhandenen Labels während der Bereitstellung auf das Storage-Volume. Storage-Administratoren können Labels pro Storage-Pool definieren und alle Volumes gruppieren, die in einem Storage-Pool erstellt wurden. Dies bietet eine praktische Möglichkeit, Volumes anhand einer Reihe anpassbarer Etiketten, die in der Backend-Konfiguration bereitgestellt werden, zu unterscheiden.

Beispiel 1: Minimale Konfiguration

Dies ist die absolute minimale Backend-Konfiguration. Mit dieser Konfiguration erkennt Astra Trident alle Ihre NetApp Accounts, Kapazitäts-Pools und Subnetze, die an ANF weltweit an jedem Standort delegiert wurden, und legt die neuen Volumes zufällig auf einem davon ab.

Diese Konfiguration eignet sich ideal, wenn Sie gerade mit ANF beginnen und die Dinge ausprobieren. In der Praxis möchten Sie jedoch zusätzliche Informationen für die Volumes bereitstellen, die Sie bereitstellen.

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
    "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
    "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
    "clientSecret": "SECRET"
}
```

Beispiel 2: Einzelner Standort und spezifische Service Level-Konfiguration

Bei dieser Back-End-Konfiguration werden Volumes in Azure platziert `eastus` Lage in A `Premium` Kapazitäts-Pool: Astra Trident erkennt automatisch alle an ANF delegierten Subnetze und legt ein neues Volume zufällig auf einen davon ab.

```
    {
        "version": 1,
        "storageDriverName": "azure-netapp-files",
        "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
        "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
        "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
        "clientSecret": "SECRET",
        "location": "eastus",
        "serviceLevel": "Premium"
    }
```

Beispiel 3: Erweiterte Konfiguration

Diese Back-End-Konfiguration reduziert den Umfang der Volume-Platzierung auf ein einzelnes Subnetz und ändert auch einige Standardwerte für die Volume-Bereitstellung.

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
    "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
    "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
    "clientSecret": "SECRET",
    "location": "eastus",
    "serviceLevel": "Premium",
    "virtualNetwork": "my-virtual-network",
    "subnet": "my-subnet",
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
    "limitVolumeSize": "500Gi",
    "defaults": {
        "exportRule": "10.0.0.0/24,10.0.1.0/24,10.0.2.100",
        "size": "200Gi"
    }
}
```

Beispiel 4: Konfiguration des virtuellen Speicherpools

Diese Back-End-Konfiguration definiert mehrere Storage-Pools in einer einzelnen Datei. Dies ist nützlich, wenn Sie über mehrere Kapazitäts-Pools verfügen, die unterschiedliche Service-Level unterstützen, und Sie Storage-Klassen in Kubernetes erstellen möchten, die diese unterstützen.

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
    "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
    "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
    "clientSecret": "SECRET",
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
    "labels": {
        "cloud": "azure"
    },
    "location": "eastus",

    "storage": [
        {
            "labels": {
                "performance": "gold"
            },
            "serviceLevel": "Ultra"
        },
        {
            "labels": {
                "performance": "silver"
            },
            "serviceLevel": "Premium"
        },
        {
            "labels": {
                "performance": "bronze"
            },
            "serviceLevel": "Standard",
        }
    ]
}
```

Im Folgenden `StorageClass` Definitionen beziehen sich auf die oben genannten Speicherpools. Durch Verwendung des `parameters.selector` Feld können Sie für jedes Feld angeben `StorageClass` Der virtuelle Pool, der zum Hosten eines Volumes genutzt wird. Im Volume werden die Aspekte definiert, die im ausgewählten Pool definiert sind.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze"
allowVolumeExpansion: true
```

Was kommt als Nächstes?

Führen Sie nach dem Erstellen der Back-End-Konfigurationsdatei den folgenden Befehl aus:

```
tridentctl create backend -f <backend-file>
```

Wenn die Backend-Erstellung fehlschlägt, ist mit der Back-End-Konfiguration ein Fehler aufgetreten. Sie können die Protokolle zur Bestimmung der Ursache anzeigen, indem Sie den folgenden Befehl ausführen:

```
tridentctl logs
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und korrigiert haben, können Sie den Befehl „Erstellen“ erneut ausführen.

Konfiguration eines CVS für AWS Backend

Erfahren Sie, wie Sie NetApp Cloud Volumes Service (CVS) für AWS mit den bereitgestellten

Beispielkonfigurationen als Backend für Ihre Astra Trident Installation konfigurieren.

> Cloud Volumes Service für AWS unterstützt keine Volumes unter 100 GB. Trident erstellt automatisch 100-GB-Volumes, wenn ein kleineres Volume angefordert wird.

Was Sie benötigen

Um den zu konfigurieren und zu verwenden "Cloud Volumes Service für AWS" Back-End, Sie benötigen Folgendes:

- Ein AWS Konto, der mit NetApp CVS konfiguriert ist
- API-Region, URL und Schlüssel für Ihr CVS-Konto

Back-End-Konfigurationsoptionen

Die Back-End-Konfigurationsoptionen finden Sie in der folgenden Tabelle:

Parameter	Beschreibung	Standard
`version`		Immer 1
`storageDriverName`	Name des Speichertreibers	„aws-cvs“
`backendName`	Benutzerdefinierter Name oder das Storage-Backend	Treibername + „_“ + Teil des API-Schlüssels
`apiRegion`	CVS-Account-Region. Den Wert finden Sie im CVS-Webportal unter Kontoeinstellungen/API-Zugriff.	
`apiURL`	CVS-KONTO-API-URL. Den Wert finden Sie im CVS-Webportal unter Kontoeinstellungen/API-Zugriff.	
`apiKey`	CVS-Konto-API-Schlüssel Den Wert finden Sie im CVS-Webportal unter Kontoeinstellungen/API-Zugriff.	
`secretKey`	Geheimer Schlüssel für CVS-Konto. Den Wert finden Sie im CVS-Webportal unter Kontoeinstellungen/API-Zugriff.	
`proxyURL`	Proxy-URL, wenn Proxyserver für die Verbindung mit CVS-Konto benötigt wird. Der Proxy-Server kann entweder ein HTTP-Proxy oder ein HTTPS-Proxy sein. Bei einem HTTPS-Proxy wird die Zertifikatvalidierung übersprungen, um die Verwendung von selbstsignierten Zertifikaten im Proxyserver zu ermöglichen. Proxy-Server mit aktivierter Authentifizierung werden nicht unterstützt.	

Parameter	Beschreibung	Standard
`nfsMountOptions`	Engmaschige Kontrolle der NFS-Mount-Optionen	„Nfsvers=3“
`limitVolumeSize`	Bereitstellung fehlgeschlagen, wenn die angeforderte Volume-Größe über diesem Wert liegt	„“ (nicht standardmäßig durchgesetzt)
`serviceLevel`	CVS Service-Level für neue Volumes Die Werte sind „Standard“, „Premium“ und „extrem“.	„Standard“
`debugTraceFlags`	Fehler-Flags bei der Fehlerbehebung beheben. Beispiel: `\{"api":false, "method":true}`. Verwenden Sie dies nur, wenn Sie Fehler beheben und einen detaillierten Log Dump benötigen.	Null

`apiURL` Ist für jedes einzigartig `apiRegion`. Zum Beispiel die US-West-2 `apiRegion` Weist den auf https://cv.us-west-2.netapp.com:8080/v1/ `apiURL`. Ebenso die US-Ost-1 `apiRegion` Weist den auf https://cds-aws-bundles.netapp.com:8080/v1/ `apiURL`. Überprüfen Sie das CVS Dashboard auf die richtige `apiRegion` Und `apiURL` Parameter für Ihre Backend-Konfiguration.

Jedes Back-End stellt Volumes in einer einzelnen AWS Region bereit. Um Volumes in anderen Regionen zu erstellen, können Sie zusätzliche Back-Ends definieren.

Sie können festlegen, wie jedes Volume standardmäßig bereitgestellt wird, indem Sie die folgenden Optionen in einem speziellen Abschnitt der Konfigurationsdatei angeben. Sehen Sie sich die Konfigurationsbeispiele unten an.

Parameter	Beschreibung	Standard
`exportRule`	Die Exportregel(n) für neue Volumes	„0.0.0.0/0“
`snapshotDir`	Steuert die Visibilität des `.snapshot` Verzeichnis	„Falsch“
`snapshotReserve`	Prozentsatz des für Snapshots reservierten Volumes	"" (CVS Standard 0 akzeptieren)
`size`	Die Größe neuer Volumes	„100 GB“

Der `exportRule` Wert muss eine kommagetrennte Liste beliebiger Kombinationen von IPv4-Adressen oder IPv4-Subnetzen in CIDR-Notation sein.

Astra Trident kopiert bei allen Volumes, die auf einem CVS AWS Backend erstellt wurden, alle auf einem Storage-Pool vorhandenen Labels zum Zeitpunkt der Bereitstellung auf das Storage-Volume. Storage-Administratoren können Labels pro Storage-Pool definieren und alle Volumes gruppieren, die in einem Storage-Pool erstellt wurden. Dies bietet eine praktische Möglichkeit, Volumes anhand einer Reihe anpassbarer Etiketten, die in der Backend-Konfiguration bereitgestellt werden, zu unterscheiden.

Beispiel 1: Minimale Konfiguration

Dies ist die absolute minimale Backend-Konfiguration.

Diese Konfiguration eignet sich ideal, wenn Sie lediglich mit CVS AWS beginnen und die Dinge ausprobieren möchten. In der Praxis möchten Sie jedoch zusätzliche Umfang für die Volumes bereitstellen, die Sie bereitstellen.

```
{
    "version": 1,
    "storageDriverName": "aws-cvs",
    "apiRegion": "us-east-1",
    "apiURL": "https://cds-aws-bundles.netapp.com:8080/v1",
    "apiKey": "znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE",
    "secretKey": "rR0rUmWXfNioN1KhtHisiSAnoTherboGuskey6pU"
}
```

Beispiel 2: Single Service Level-Konfiguration

Dieses Beispiel zeigt eine Backend-Datei, die dieselben Aspekte auf allen mit Astra Trident erstellten Storage in der Region AWS US-East-1 anwendet. In diesem Beispiel wird auch die Verwendung von angezeigt `proxyURL` In der Backend-Datei.

```
{
    "version": 1,
    "storageDriverName": "aws-cvs",
    "backendName": "cvs-aws-us-east",
    "apiRegion": "us-east-1",
    "apiURL": "https://cds-aws-bundles.netapp.com:8080/v1",
    "apiKey": "znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE",
    "secretKey": "rR0rUmWXfNioN1KhtHisiSAnoTherboGuskey6pU",
    "proxyURL": "http://proxy-server-hostname/",
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
    "limitVolumeSize": "50Gi",
    "serviceLevel": "premium",
    "defaults": {
        "snapshotDir": "true",
        "snapshotReserve": "5",
        "exportRule": "10.0.0.0/24,10.0.1.0/24,10.0.2.100",
        "size": "200Gi"
    }
}
```

Beispiel 3: Konfiguration des virtuellen Speicherpools

Dieses Beispiel zeigt die mit virtuellen Speicherpools und StorageClasses konfigurierte Back-End-Definitionsdatei.

In der unten gezeigten Beispiel-Backend-Definitionsdatei werden für alle Speicherpools spezifische Standardwerte festgelegt, die die definieren `snapshotReserve` Bei 5% und der `exportRule` Zu 0.0.0.0/0. Die virtuellen Speicherpools werden im definiert `storage` Abschnitt. In diesem Beispiel legt jeder einzelne Storage-Pool seinen eigenen fest `serviceLevel`, Und einige Pools überschreiben die Standardwerte.

```
{
    "version": 1,
    "storageDriverName": "aws-cvs",
    "apiRegion": "us-east-1",
    "apiURL": "https://cds-aws-bundles.netapp.com:8080/v1",
    "apiKey": "EnterYourAPIKeyHere**********************",
    "secretKey": "EnterYourSecretKeyHere******************",
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",

    "defaults": {
        "snapshotReserve": "5",
        "exportRule": "0.0.0.0/0"
    },

    "labels": {
        "cloud": "aws"
```

```
    },
    "region": "us-east-1",

    "storage": [
        {
            "labels": {
                "performance": "extreme",
                "protection": "extra"
            },
            "serviceLevel": "extreme",
            "defaults": {
                "snapshotDir": "true",
                "snapshotReserve": "10",
                "exportRule": "10.0.0.0/24"
            }
        },
        {
            "labels": {
                "performance": "extreme",
                "protection": "standard"
            },
            "serviceLevel": "extreme"
        },
        {
            "labels": {
                "performance": "premium",
                "protection": "extra"
            },
            "serviceLevel": "premium",
            "defaults": {
                "snapshotDir": "true",
                "snapshotReserve": "10"
            }
        },

        {
            "labels": {
                "performance": "premium",
                "protection": "standard"
            },
            "serviceLevel": "premium"
        },

        {
            "labels": {
                "performance": "standard"
```

```
            },
            "serviceLevel": "standard"
        }
    ]
}
```

Die folgenden StorageClass-Definitionen beziehen sich auf die oben genannten Speicherpools. Durch Verwendung des `parameters.selector` Feld können Sie für jede StorageClass den virtuellen Pool angeben, der zum Hosten eines Volumes verwendet wird. Im Volume werden die Aspekte definiert, die im ausgewählten Pool definiert sind.

Die erste StorageClass (`cvs-extreme-extra-protection`) Zuordnung zum ersten virtuellen Speicherpool. Dies ist der einzige Pool, der eine extreme Performance mit einer Snapshot-Reserve von 10 % bietet. Die letzte StorageClass (`cvs-extra-protection`) Ruft alle Speicher-Pool, die eine Snapshot-Reserve von 10% bietet. Astra Trident entscheidet, welcher Virtual Storage Pool ausgewählt wird und stellt sicher, dass die Anforderungen an die Snapshot-Reserve erfüllt werden.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=extreme; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=extreme; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
```

```
  name: cvs-premium
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
provisioner: netapp.io/trident
parameters:
  selector: "performance=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "protection=extra"
allowVolumeExpansion: true
```

Was kommt als Nächstes?

Führen Sie nach dem Erstellen der Back-End-Konfigurationsdatei den folgenden Befehl aus:

```
tridentctl create backend -f <backend-file>
```

Wenn die Backend-Erstellung fehlschlägt, ist mit der Back-End-Konfiguration ein Fehler aufgetreten. Sie können die Protokolle zur Bestimmung der Ursache anzeigen, indem Sie den folgenden Befehl ausführen:

```
tridentctl logs
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und korrigiert haben, können Sie den Befehl „Erstellen“ erneut ausführen.

Konfiguration eines CVS für GCP-Backend

Erfahren Sie, wie Sie NetApp Cloud Volumes Service (CVS) für die Google Cloud Platform (GCP) als Backend für Ihre Astra Trident Installation mit den angegebenen Beispielkonfigurationen konfigurieren.

NetApp Cloud Volumes Service für Google Cloud unterstützt keine CVS-Performance Volumes mit einer Größe von weniger als 100 gib oder CVS Volumes mit einer Größe von weniger als 300 gib. Astra Trident erstellt automatisch Volumes mit der Mindestgröße, wenn das angeforderte Volume kleiner als die Mindestgröße ist.

Was Sie benötigen

Um den zu konfigurieren und zu verwenden "Cloud Volumes Service für Google Cloud" Back-End, Sie benötigen Folgendes:

- Ein Google Cloud Konto, der mit NetApp CVS konfiguriert ist
- Projektnummer Ihres Google Cloud-Kontos
- Google Cloud-Servicekonto bei `netappcloudvolumes.admin` Rolle
- API-Schlüsseldatei für Ihr CVS-Servicekonto

Astra Trident unterstützt jetzt auch kleinere Volumes – standardmäßig "Servicetypen":[CVS-Diensttyp auf GCP^]. Für mit erstellte Back-Ends `storageClass=software`, Volumes verfügen jetzt über eine minimale Bereitstellungsgröße von 300 gib. CVS bietet diese Funktion derzeit unter Controlled Availability und bietet keinen technischen Support. Benutzer müssen sich für den Zugriff auf Sub-1-tib-Volumes anmelden "Hier". NetApp empfiehlt Kunden, Sub-1-tib-Volumes für **nicht produktive**-Workloads zu nutzen.

Bei der Bereitstellung von Back-Ends mithilfe des standardmäßigen CVS-Servicetyps (`storageClass=software`), Benutzer müssen Zugriff auf die Sub-1-tib-Volumes-Funktion in GCP für die Projektnummer(n) und Projekt-ID(s) in Frage erhalten. Dieser ist für die Bereitstellung von Sub-1-tib-Volumes durch Astra Trident erforderlich. Wenn nicht, schlagen Volumenkreationen bei VES fehl, die weniger als 600 gib betragen. Zugriff auf Sub-1-tib-Volumes mit "Dieses Formular".

Von Astra Trident erstellte Volumes für den CVS Standard-Service-Level werden wie folgt bereitgestellt:

- PVCs, die kleiner als 300 gib sind, führen dazu, dass Astra Trident ein CVS-Volumen von 300 gib erstellt.
- PVCs, die zwischen 300 gib und 600 gib liegen, führen dazu, dass Astra Trident ein CVS-Volumen der angeforderten Größe erstellt.
- PVCs, die zwischen 600 gib und 1 tib liegen, führen dazu, dass Astra Trident ein CVS Volume mit 1 tib erstellt.
- PVCs, die mehr als 1 tib sind, führen dazu, dass Astra Trident ein CVS Volume der angeforderten Größe erstellt.

Back-End-Konfigurationsoptionen

Die Back-End-Konfigurationsoptionen finden Sie in der folgenden Tabelle:

Parameter	Beschreibung	Standard
`version`		Immer 1
`storageDriverName`	Name des Speichertreibers	„gcp-cvs“
`backendName`	Benutzerdefinierter Name oder das Storage-Backend	Treibername + „_“ + Teil des API-Schlüssels

Parameter	Beschreibung	Standard
`storageClass`	Art des Storage: Wählen Sie aus `hardware` (Performance optimiert) oder `software` (CVS-Servicetyp)	
`projectNumber`	Google Cloud Account Projektnummer. Der Wert ist auf der Homepage des Google Cloud Portals zu finden.	
`apiRegion`	CVS-Account-Region. Es ist der Bereich, in dem das Backend die Volumen bereitstellen wird.	
`apiKey`	API-Schlüssel für das Google Cloud-Dienstkonto bei `netappcloudvolumes.admin` Rolle: Er enthält den JSON-formatierten Inhalt der privaten Schlüsseldatei eines Google Cloud-Dienstkontos (wortgetreu in die Back-End-Konfigurationsdatei kopiert).	
`proxyURL`	Proxy-URL, wenn Proxyserver für die Verbindung mit CVS-Konto benötigt wird. Der Proxy-Server kann entweder ein HTTP-Proxy oder ein HTTPS-Proxy sein. Bei einem HTTPS-Proxy wird die Zertifikatvalidierung übersprungen, um die Verwendung von selbstsignierten Zertifikaten im Proxyserver zu ermöglichen. Proxy-Server mit aktivierter Authentifizierung werden nicht unterstützt.	
`nfsMountOptions`	Engmaschige Kontrolle der NFS-Mount-Optionen	„Nfsvers=3“
`limitVolumeSize`	Bereitstellung fehlgeschlagen, wenn die angeforderte Volume-Größe über diesem Wert liegt	„“ (nicht standardmäßig durchgesetzt)
`serviceLevel`	CVS Service-Level für neue Volumes Die Werte sind „Standard“, „Premium“ und „extrem“.	„Standard“
`network`	Für CVS Volumes verwendetes GCP-Netzwerk	„Standard“

Parameter	Beschreibung	Standard
`debugTraceFlags`	Fehler-Flags bei der Fehlerbehebung beheben. Beispiel: `\{"api":false, "method":true}`. Verwenden Sie dies nur, wenn Sie Fehler beheben und einen detaillierten Log Dump benötigen.	Null

Bei der Verwendung eines gemeinsamen VPC-Netzwerks von beiden `projectNumber` Und `hostProjectNumber` Muss angegeben werden. In diesem Fall `projectNumber` Ist das Service-Projekt, und `hostProjectNumber` Ist das Hostprojekt.

Der `apiRegion` Repräsentiert die GCP-Region, in der Astra Trident CVS Volumes erstellt Astra Trident kann Volumes mounten und an Kubernetes-Nodes anhängen, die zur gleichen GCP-Region gehören. Bei der Erstellung eines Backend ist es wichtig, sicherzustellen `apiRegion` Stimmt mit der Region überein, in werden Kubernetes-Nodes implementiert. Wenn über mehrere Regionen hinweg Kubernetes Cluster erstellt werden, werden CVS Volumes in einem bestimmten erstellt `apiRegion` Kann nur in Workloads verwendet werden, die für Nodes geplant sind, die sich in derselben GCP-Region befinden.

`storageClass` Ist ein optionaler Parameter, mit dem Sie das gewünschte auswählen können "CVS-Diensttyp". Sie haben die Wahl zwischen dem Basistyp CVS (`storageClass=software`) Oder den Servicetyp CVS-Performance (`storageClass=hardware`), die Trident standardmäßig verwendet. Stellen Sie sicher, dass Sie ein angeben `apiRegion` Das bietet das jeweilige CVS `storageClass` Back-End-Definition:

Die Integration von Astra Trident mit dem Basis-CVS-Servicetyp auf Google Cloud ist eine **Beta-Funktion**, die nicht für Produktions-Workloads bestimmt ist. Trident wird **vollständig unterstützt** mit dem Service-Typ CVS-Performance und verwendet ihn standardmäßig.

Jedes Back-End stellt Volumes in einer einzigen Google Cloud-Region bereit. Um Volumes in anderen Regionen zu erstellen, können Sie zusätzliche Back-Ends definieren.

Sie können festlegen, wie jedes Volume standardmäßig bereitgestellt wird, indem Sie die folgenden Optionen in einem speziellen Abschnitt der Konfigurationsdatei angeben. Sehen Sie sich die Konfigurationsbeispiele unten an.

Parameter	Beschreibung	Standard
`exportRule`	Die Exportregel(n) für neue Volumes	„0.0.0.0/0“
`snapshotDir`	Zugriff auf die `.snapshot` Verzeichnis	„Falsch“
`snapshotReserve`	Prozentsatz des für Snapshots reservierten Volumes	"" (CVS Standard 0 akzeptieren)
`size`	Die Größe neuer Volumes	„100 Gi“

Der `exportRule` Wert muss eine kommagetrennte Liste beliebiger Kombinationen von IPv4-Adressen oder IPv4-Subnetzen in CIDR-Notation sein.

Trident kopiert bei allen Volumes, die auf einem Google Cloud Backend von CVS erstellt wurden, alle auf einem Storage-Pool vorhandenen Labels zum Zeitpunkt der Bereitstellung auf das Storage-Volume. Storage-Administratoren können Labels pro Storage-Pool definieren und alle Volumes gruppieren, die in einem Storage-Pool erstellt wurden. Dies bietet eine praktische Möglichkeit, Volumes anhand einer Reihe anpassbarer Etiketten, die in der Backend-Konfiguration bereitgestellt werden, zu unterscheiden.

Beispiel 1: Minimale Konfiguration

Dies ist die absolute minimale Backend-Konfiguration.

```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "apiRegion": "us-west2",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "1234567890123456789012345678901234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisI
sAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSa
PIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZN
chRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlz
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHi
sIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOgu
SaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyA
ZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz
lzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrt
HisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbO
guSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKe
yAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRA
GzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
```

```
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    }
}
```

Beispiel 2: Konfiguration des Basis-CVS-Diensttyps

Dieses Beispiel zeigt eine Backend-Definition, die den CVS Basis-Service-Typ nutzt, der für allgemeine Workloads gedacht ist und eine geringe/mittlere Performance bietet, sowie eine hohe zonale Verfügbarkeit.

```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "storageClass": "software",
    "apiRegion": "us-east4",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "1234567890123456789012345678901234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisI
sAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSa
PIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZN
chRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlz
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHi
sIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOgu
SaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyA
ZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz
lzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrt
HisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbO
```

```
guSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKe
yAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRA
GzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    }
}
```

Beispiel 3: Einzel-Service Level-Konfiguration

Dieses Beispiel zeigt eine Backend-Datei, die dieselben Aspekte auf allen mit Astra Trident erstellten Storage in der Region Google Cloud US-west2 anwendet. In diesem Beispiel wird auch die Verwendung von angezeigt `proxyURL` In der Back-End-Konfigurationsdatei

```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "apiRegion": "us-west2",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "1234567890123456789012345678901234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisI
sAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSa
PIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZN
chRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlz
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHi
sIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOgu
SaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyA
```

```
ZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz
lzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrt
HisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbO
guSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKe
yAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRA
GzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    },
    "proxyURL": "http://proxy-server-hostname/",
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
    "limitVolumeSize": "10Ti",
    "serviceLevel": "premium",
    "defaults": {
        "snapshotDir": "true",
        "snapshotReserve": "5",
        "exportRule": "10.0.0.0/24,10.0.1.0/24,10.0.2.100",
        "size": "5Ti"
    }
}
```

Beispiel 4: Konfiguration des virtuellen Speicherpools

Dieses Beispiel zeigt die Back-End-Definitionsdatei, die mit virtuellen Speicherpools konfiguriert ist `StorageClasses` Die sich auf sie beziehen.

In der unten gezeigten Beispiel-Backend-Definitionsdatei werden für alle Speicherpools spezifische Standardwerte festgelegt, die die definieren `snapshotReserve` Bei 5% und der `exportRule` Zu 0.0.0.0/0. Die virtuellen Speicherpools werden im definiert `storage` Abschnitt. In diesem Beispiel legt jeder einzelne Storage-Pool seinen eigenen fest `serviceLevel`, Und einige Pools überschreiben die Standardwerte.

```
{
```

```
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "apiRegion": "us-west2",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "1234567890123456789012345678901234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisI
sAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSa
PIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZN
chRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlz
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHi
sIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOgu
SaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyA
ZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz
lzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrt
HisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbO
guSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKe
yAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRA
GzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    },
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
```

```
    "defaults": {
        "snapshotReserve": "5",
        "exportRule": "0.0.0.0/0"
    },

    "labels": {
        "cloud": "gcp"
    },
    "region": "us-west2",

    "storage": [
        {
            "labels": {
                "performance": "extreme",
                "protection": "extra"
            },
            "serviceLevel": "extreme",
            "defaults": {
                "snapshotDir": "true",
                "snapshotReserve": "10",
                "exportRule": "10.0.0.0/24"
            }
        },
        {
            "labels": {
                "performance": "extreme",
                "protection": "standard"
            },
            "serviceLevel": "extreme"
        },
        {
            "labels": {
                "performance": "premium",
                "protection": "extra"
            },
            "serviceLevel": "premium",
            "defaults": {
                "snapshotDir": "true",
                "snapshotReserve": "10"
            }
        },

        {
            "labels": {
                "performance": "premium",
                "protection": "standard"
```

```
            },
            "serviceLevel": "premium"
        },

        {
            "labels": {
                "performance": "standard"
            },
            "serviceLevel": "standard"
        }
    ]
}
```

Die folgenden StorageClass-Definitionen beziehen sich auf die oben genannten Speicherpools. Durch Verwendung des `parameters.selector` Feld können Sie für jede StorageClass den virtuellen Pool angeben, der zum Hosten eines Volumes verwendet wird. Im Volume werden die Aspekte definiert, die im ausgewählten Pool definiert sind.

Die erste StorageClass (`cvs-extreme-extra-protection`) Zuordnung zum ersten virtuellen Speicherpool. Dies ist der einzige Pool, der eine extreme Performance mit einer Snapshot-Reserve von 10 % bietet. Die letzte StorageClass (`cvs-extra-protection`) Ruft alle Speicher-Pool, die eine Snapshot-Reserve von 10% bietet. Astra Trident entscheidet, welcher Virtual Storage Pool ausgewählt wird und stellt sicher, dass die Anforderungen an die Snapshot-Reserve erfüllt werden.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=extreme; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: netapp.io/trident
```

```
parameters:
  selector: "performance=premium; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
provisioner: netapp.io/trident
parameters:
  selector: "performance=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "protection=extra"
allowVolumeExpansion: true
```

Was kommt als Nächstes?

Führen Sie nach dem Erstellen der Back-End-Konfigurationsdatei den folgenden Befehl aus:

```
tridentctl create backend -f <backend-file>
```

Wenn die Backend-Erstellung fehlschlägt, ist mit der Back-End-Konfiguration ein Fehler aufgetreten. Sie können die Protokolle zur Bestimmung der Ursache anzeigen, indem Sie den folgenden Befehl ausführen:

```
tridentctl logs
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und korrigiert haben, können Sie den Befehl „Erstellen“ erneut ausführen.

Konfigurieren Sie ein NetApp HCI- oder SolidFire-Backend

Erfahren Sie, wie Sie mit Ihrer Astra Trident Installation ein Element Backend erstellen und verwenden.

Was Sie benötigen

- Ein unterstütztes Storage-System, auf dem die Element Software ausgeführt wird.
- Anmeldedaten für einen NetApp HCI/SolidFire Cluster-Administrator oder einen Mandantenbenutzer, der Volumes managen kann
- Alle Kubernetes-Worker-Nodes sollten die entsprechenden iSCSI-Tools installiert haben. Siehe "Informationen zur Vorbereitung auf den Worker-Node".

Was Sie wissen müssen

Der `solidfire-san` Der Storage-Treiber unterstützt beide Volume-Modi: Datei und Block. Für das `Filesystem` VolumeMode erstellt Astra Trident ein Volume und erstellt ein Dateisystem. Der Dateisystem-Typ wird von StorageClass angegeben.

Treiber	Protokoll	VolumeMode	Unterstützte Zugriffsmodi	Unterstützte Filesysteme
`solidfire-san`	ISCSI	Block-Storage	RWO, ROX, RWX	Kein Dateisystem. Rohes Blockgerät.
`solidfire-san`	ISCSI	Block-Storage	RWO, ROX, RWX	Kein Dateisystem. Rohes Blockgerät.
`solidfire-san`	ISCSI	Dateisystem	RWO, ROX	`xfs`, `ext3`, `ext4`
`solidfire-san`	ISCSI	Dateisystem	RWO, ROX	`xfs`, `ext3`, `ext4`

Astra Trident verwendet CHAP, wenn es als erweiterte CSI-Bereitstellung funktioniert. Wenn Sie CHAP verwenden (das ist die Standardeinstellung für CSI), ist keine weitere Vorbereitung erforderlich. Es wird empfohlen, das explizit festzulegen `UseCHAP` Option zur Verwendung von CHAP mit nicht-CSI Trident. Anderenfalls siehe "Hier".

Volume-Zugriffsgruppen werden nur vom herkömmlichen, nicht-CSI-Framework für Astra Trident unterstützt. Bei der Konfiguration für die Verwendung im CSI-Modus verwendet Astra Trident CHAP.

Wenn keine `AccessGroups` Oder `UseCHAP` Sind festgelegt, gilt eines der folgenden Regeln:

- Wenn die Standardeinstellung `trident` Zugriffsgruppe wird erkannt, Zugriffsgruppen werden verwendet.
- Wenn keine Zugriffsgruppe erkannt wird und die Kubernetes-Version 1.7 oder höher ist, wird CHAP verwendet.

Back-End-Konfigurationsoptionen

Die Back-End-Konfigurationsoptionen finden Sie in der folgenden Tabelle:

Parameter	Beschreibung	Standard
`version`		Immer 1
`storageDriverName`	Name des Speichertreibers	Immer „solidfire-san“
`backendName`	Benutzerdefinierter Name oder das Storage-Backend	IP-Adresse „SolidFire_“ + Storage (iSCSI)
`Endpoint`	MVIP für den SolidFire-Cluster mit Mandanten-Anmeldedaten	
`SVIP`	Speicher-IP-Adresse und -Port	
`labels`	Satz willkürlicher JSON-formatierter Etiketten für Volumes.	„“
`TenantName`	Zu verwendende Mandantenbezeichnung (wird erstellt, wenn sie nicht gefunden wurde)	
`InitiatorIFace`	Beschränken Sie den iSCSI-Datenverkehr auf eine bestimmte Host-Schnittstelle	„Standard“
`UseCHAP`	Verwenden Sie CHAP, um iSCSI zu authentifizieren	Richtig
`AccessGroups`	Liste der zu verwendenden Zugriffsgruppen-IDs	Findet die ID einer Zugriffsgruppe namens „Dreizack“
`Types`	QoS-Spezifikationen	
`limitVolumeSize`	Bereitstellung fehlgeschlagen, wenn die angeforderte Volume-Größe über diesem Wert liegt	„“ (nicht standardmäßig durchgesetzt)
`debugTraceFlags`	Fehler-Flags bei der Fehlerbehebung beheben. Beispiel: { „API“:false, „Methode“:true}	Null

Verwenden Sie es nicht `debugTraceFlags` Es sei denn, Sie beheben Fehler und benötigen einen detaillierten Log Dump.

Astra Trident kopiert bei allen erstellten Volumes alle auf einem Storage-Pool vorhandenen Beschriftungen in die zugrunde liegende Storage-LUN zum Zeitpunkt der Bereitstellung. Storage-Administratoren können Labels pro Storage-Pool definieren und alle Volumes gruppieren, die in einem Storage-Pool erstellt wurden. Dies bietet eine praktische Möglichkeit, Volumes anhand einer Reihe anpassbarer Etiketten, die in der Backend-Konfiguration bereitgestellt werden, zu unterscheiden.

Beispiel 1: Back-End-Konfiguration für `solidfire-san` Treiber mit drei Lautstärketypen

Dieses Beispiel zeigt eine Backend-Datei mit CHAP-Authentifizierung und Modellierung von drei Volume-Typen mit spezifischen QoS-Garantien. Sehr wahrscheinlich würden Sie dann Storage-Klassen definieren, um jeden davon mit dem zu nutzen `IOPS` Parameter für Storage-Klasse.

```
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://<user>:<password>@<mvip>/json-rpc/8.0",
    "SVIP": "<svip>:3260",
    "TenantName": "<tenant>",
    "labels": {"k8scluster": "dev1", "backend": "dev1-element-cluster"},
    "UseCHAP": true,
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS": 2000,
"burstIOPS": 4000}},
             {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS": 6000,
"burstIOPS": 8000}},
             {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS": 8000,
"burstIOPS": 10000}}]
}
```

Beispiel 2: Back-End- und Storage-Class-Konfiguration für `solidfire-san` Treiber mit virtuellen Speicherpools

Dieses Beispiel zeigt die mit virtuellen Speicherpools und StorageClasses konfigurierte Back-End-Definitionsdatei.

In der unten gezeigten Beispiel-Backend-Definitionsdatei werden für alle Speicherpools spezifische Standardwerte festgelegt, die die definieren `type` Bei Silver. Die virtuellen Speicherpools werden im definiert `storage` Abschnitt. In diesem Beispiel legt ein Teil des Speicherpools seinen eigenen Typ fest, und einige Pools überschreiben die oben festgelegten Standardwerte.

```
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://<user>:<password>@<mvip>/json-rpc/8.0",
    "SVIP": "<svip>:3260",
    "TenantName": "<tenant>",
    "UseCHAP": true,
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS": 2000,
"burstIOPS": 4000}},
             {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS": 6000,
"burstIOPS": 8000}},
             {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS": 8000,
"burstIOPS": 10000}}],

    "type": "Silver",
    "labels":{"store":"solidfire", "k8scluster": "dev-1-cluster"},
    "region": "us-east-1",

    "storage": [
        {
            "labels":{"performance":"gold", "cost":"4"},
            "zone":"us-east-1a",
            "type":"Gold"
        },
        {
            "labels":{"performance":"silver", "cost":"3"},
            "zone":"us-east-1b",
            "type":"Silver"
        },
        {
            "labels":{"performance":"bronze", "cost":"2"},
            "zone":"us-east-1c",
            "type":"Bronze"
        },
        {
            "labels":{"performance":"silver", "cost":"1"},
            "zone":"us-east-1d"
        }
    ]
}
```

Die folgenden StorageClass-Definitionen beziehen sich auf die oben genannten virtuellen Speicherpools. Verwenden der `parameters.selector` Feld gibt in jeder StorageClass an, welche virtuellen Pools zum Hosten eines Volumes verwendet werden können. Auf dem Volume werden die Aspekte im ausgewählten virtuellen Pool definiert.

Die erste StorageClass (`solidfire-gold-four`) Wird dem ersten virtuellen Speicherpool zugeordnet. Dies ist der einzige Pool, der Gold Performance mit einem bietet `Volume Type QoS` Von Gold. Die letzte StorageClass (`solidfire-silver`) Bezeichnet jeden Speicherpool, der eine silberne Leistung bietet. Astra Trident entscheidet, welcher virtuelle Storage Pool ausgewählt wird und ob die Storage-Anforderungen erfüllt werden.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold; cost=4"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=3"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze; cost=2"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=1"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
  fsType: "ext4"
```

Weitere Informationen

- "Volume-Zugriffsgruppen"

Konfigurieren Sie ein Backend mit ONTAP-SAN-Treibern

Erfahren Sie mehr über die Konfiguration eines ONTAP-Backends mit ONTAP-SAN-Treibern.

- "Vorbereitung"
- "Konfiguration und Beispiele"

Benutzerberechtigungen

Astra Trident erwartet, dass er entweder als ONTAP- oder SVM-Administrator ausgeführt wird, in der Regel mit dem `admin` Cluster-Benutzer oder ein `vsadmin` SVM-Benutzer oder ein Benutzer mit einem anderen Namen und derselben Rolle. Astra Trident erwartet, dass bei Amazon FSX für Implementierungen von NetApp ONTAP, über das Cluster entweder als ONTAP- oder SVM-Administrator ausgeführt wird `fsxadmin` Benutzer oder A `vsadmin` SVM-Benutzer oder ein Benutzer mit einem anderen Namen und derselben Rolle. Der `fsxadmin` Der Benutzer ist ein eingeschränkter Ersatz für den Cluster-Admin-Benutzer.

> Wenn Sie den verwenden `limitAggregateUsage` Parameter, Berechtigungen für Cluster-Admin sind erforderlich. Bei der Verwendung von Amazon FSX für NetApp ONTAP mit Astra Trident, das `limitAggregateUsage` Der Parameter funktioniert nicht mit dem `vsadmin` Und `fsxadmin` Benutzerkonten. Der Konfigurationsvorgang schlägt fehl, wenn Sie diesen Parameter angeben.

Vorbereitung

Erfahren Sie, wie Sie ein ONTAP-Back-End mit ONTAP-SAN-Treibern vorbereiten. Für alle ONTAP Back-Ends benötigt Astra Trident mindestens ein Aggregat, das der SVM zugewiesen ist.

Denken Sie daran, dass Sie auch mehr als einen Treiber ausführen können und Speicherklassen erstellen können, die auf den einen oder anderen verweisen. Beispielsweise könnten Sie A konfigurieren `san-dev` Klasse, die den verwendet `ontap-san` Fahrer und A `san-default` Klasse, die den verwendet `ontap-san-economy` Eins.

Alle Kubernetes-Worker-Nodes müssen über die entsprechenden iSCSI-Tools verfügen. Siehe "Hier" Entnehmen.

Authentifizierung

Astra Trident bietet zwei Arten der Authentifizierung eines ONTAP-Backend.

- Anmeldeinformationsbasiert: Benutzername und Passwort für einen ONTAP-Benutzer mit den erforderlichen Berechtigungen. Es wird empfohlen, eine vordefinierte Sicherheits-Login-Rolle zu verwenden, wie z. B. `admin` Oder `vsadmin` Für maximale Kompatibilität mit ONTAP Versionen.
- Zertifikatsbasiert: Astra Trident kann auch mit einem ONTAP Cluster kommunizieren. Verwenden Sie dazu ein Zertifikat, das auf dem Backend installiert ist. Hier muss die Backend-Definition Base64-kodierte Werte des Client-Zertifikats, des Schlüssels und des vertrauenswürdigen CA-Zertifikats enthalten, sofern verwendet (empfohlen).

Benutzer können auch vorhandene Back-Ends aktualisieren, sich für die Migration von Anmeldeinformationsbasierten zu zertifikatbasierten Optionen entscheiden und umgekehrt. Wenn **sowohl**

Anmeldeinformationen als auch Zertifikate bereitgestellt werden, verwendet Astra Trident standardmäßig Zertifikate, während eine Warnung ausgegeben wird, um die Anmeldeinformationen aus der Back-End-Definition zu entfernen.

Aktivieren Sie die Anmeldeinformationsbasierte Authentifizierung

Astra Trident erfordert die Zugangsdaten für einen Administrator mit SVM-Umfang/Cluster-Umfang, um mit dem Backend von ONTAP zu kommunizieren. Es wird empfohlen, die Standard-vordefinierten Rollen wie zu verwenden `admin` Oder `vsadmin`. So ist gewährleistet, dass die Kompatibilität mit künftigen ONTAP Versionen gewährleistet ist, die FunktionsAPIs der künftigen Astra Trident Versionen bereitstellen können. Eine benutzerdefinierte Sicherheits-Login-Rolle kann mit Astra Trident erstellt und verwendet werden, wird aber nicht empfohlen.

Eine Beispiel-Back-End-Definition sieht folgendermaßen aus:

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "secret",
}
```

Beachten Sie, dass die Backend-Definition der einzige Ort ist, an dem die Anmeldeinformationen im reinen Text gespeichert werden. Nach der Erstellung des Backend werden Benutzernamen/Passwörter mit Base64 codiert und als Kubernetes Secrets gespeichert. Die Erstellung/Aktualisierung eines Backend ist der einzige Schritt, der Kenntnisse der Anmeldeinformationen erfordert. Daher ist dieser Vorgang nur für Administratoren und wird vom Kubernetes-/Storage-Administrator ausgeführt.

Aktivieren Sie die zertifikatbasierte Authentifizierung

Neue und vorhandene Back-Ends können ein Zertifikat verwenden und mit dem ONTAP-Back-End kommunizieren. In der Backend-Definition sind drei Parameter erforderlich.

- ClientCertificate: Base64-codierter Wert des Clientzertifikats.
- ClientPrivateKey: Base64-kodierte Wert des zugeordneten privaten Schlüssels.
- Trusted CACertificate: Base64-codierter Wert des vertrauenswürdigen CA-Zertifikats. Bei Verwendung einer vertrauenswürdigen CA muss dieser Parameter angegeben werden. Dies kann ignoriert werden, wenn keine vertrauenswürdige CA verwendet wird.

Ein typischer Workflow umfasst die folgenden Schritte.

Schritte

1. Erzeugen eines Clientzertifikats und eines Schlüssels. Legen Sie beim Generieren den allgemeinen Namen (CN) für den ONTAP-Benutzer fest, der sich authentifizieren soll als.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. Fügen Sie dem ONTAP-Cluster ein vertrauenswürdiges CA-Zertifikat hinzu. Dies kann möglicherweise bereits vom Storage-Administrator übernommen werden. Ignorieren, wenn keine vertrauenswürdige CA verwendet wird.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Installieren Sie das Client-Zertifikat und den Schlüssel (von Schritt 1) auf dem ONTAP-Cluster.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. Bestätigen Sie, dass die ONTAP-Sicherheitsanmeldungsrolle unterstützt wird `cert` Authentifizierungsmethode.

```
security login create -user-or-group-name admin -application ontapi
-authentication-method cert
security login create -user-or-group-name admin -application http
-authentication-method cert
```

5. Testen Sie die Authentifizierung mithilfe des generierten Zertifikats. <ONTAP Management LIF> und <vServer Name> durch Management-LIF-IP und SVM-Namen ersetzen.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Encodieren von Zertifikat, Schlüssel und vertrauenswürdigem CA-Zertifikat mit Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Erstellen Sie das Backend mit den Werten, die aus dem vorherigen Schritt ermittelt wurden.

```
$ cat cert-backend.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"trustedCACertificate": "QNFinfO...SiqOyN",
"storagePrefix": "myPrefix_"
}

$ tridentctl create backend -f cert-backend.json -n trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       0 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Aktualisieren Sie Authentifizierungsmethoden, oder drehen Sie die Anmeldedaten

Sie können ein vorhandenes Backend aktualisieren, um eine andere Authentifizierungsmethode zu verwenden oder um ihre Anmeldeinformationen zu drehen. Das funktioniert auf beide Arten: Back-Ends, die einen Benutzernamen/ein Passwort verwenden, können aktualisiert werden, um Zertifikate zu verwenden; Back-Ends, die Zertifikate verwenden, können auf Benutzername/Passwort-basiert aktualisiert werden. Verwenden Sie dazu ein aktualisiertes `backend.json` Datei mit den erforderlichen Parametern für die Ausführung `tridentctl backend update`.

```
$ cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "secret",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
$ tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Bei der Änderung von Passwörtern muss der Speicheradministrator das Kennwort für den Benutzer auf ONTAP aktualisieren. Auf diese Weise folgt ein Backend-Update. Beim Drehen von Zertifikaten können dem Benutzer mehrere Zertifikate hinzugefügt werden. Das Backend wird dann aktualisiert und verwendet das neue Zertifikat. Danach kann das alte Zertifikat aus dem ONTAP Cluster gelöscht werden.

Durch die Aktualisierung eines Backend wird der Zugriff auf Volumes, die bereits erstellt wurden, nicht unterbrochen, und auch die danach erstellten Volume-Verbindungen werden beeinträchtigt. Ein erfolgreiches Backend-Update zeigt, dass Astra Trident mit dem ONTAP-Backend kommunizieren und zukünftige Volume-Operationen verarbeiten kann.

Geben Sie Initiatorgruppen an

Astra Trident verwendet Initiatorgruppen, um den Zugriff auf die Volumes (LUNs) zu steuern, die er bereitstellt. Administratoren verfügen über zwei Optionen, wenn es um das Angeben von Initiatorgruppen für Back-Ends geht:

- Astra Trident kann automatisch eine igroup pro Backend erstellen und managen. Wenn `igroupName` Ist nicht in der Backend-Definition enthalten, erstellt Astra Trident eine igroup mit dem Namen `trident-<backend-UUID>` Auf der SVM. So wird sichergestellt, dass jedes Backend über eine dedizierte iGroup verfügt und das automatisierte Hinzufügen/Löschen von Kubernetes Node-IQNs behandelt.

- Alternativ können auch vorab erstellte Initiatorgruppen in einer Backend-Definition bereitgestellt werden. Dies kann mit dem erfolgen `igroupName` Konfigurationsparameter. Astra Trident fügt der bereits vorhandenen iGroup Kubernetes-Node-IQNs hinzu/löschen.

Für Back-Ends mit `igroupName` Definiert, das `igroupName` Kann mit einem gelöscht werden `tridentctl backend update` Astra Trident ist die Auto-Handle-Initiatorgruppen. Dadurch wird der Zugriff auf Volumes nicht unterbrochen, die bereits an Workloads angeschlossen sind. Künftige Verbindungen werden mit der von der igroup Astra Trident erstellten iGroup behandelt.

> Die Einwidmung einer Initiatorgruppe für jede einzelne Instanz des Astra Trident ist eine Best Practice, die sowohl dem Kubernetes-Administrator als auch dem Storage-Administrator von Vorteil ist. CSI Trident automatisiert das Hinzufügen und Entfernen von Cluster Node-IQNs zur igroup und vereinfacht das Management enorm. Wenn in Kubernetes-Umgebungen dieselben SVMs verwendet werden (und Astra Trident-Installationen), stellt die Verwendung einer dedizierten igroup sicher, dass Änderungen an einem Kubernetes-Cluster keinen Einfluss auf Initiatorgruppen haben, die anderen zugeordnet sind. Darüber hinaus ist es wichtig, dass jeder Node im Kubernetes Cluster über einen eindeutigen IQN verfügt. Wie oben erwähnt, übernimmt Astra Trident automatisch das Hinzufügen und Entfernen von IQNs. Die Wiederverwendung von IQNs über Hosts kann zu unerwünschten Szenarien führen, in denen Hosts sich gegenseitig irren und der Zugriff auf LUNs verweigert wird.

Wenn Astra Trident als CSI-Bereitstellung konfiguriert ist, werden Kubernetes-Node-IQNs automatisch der Initiatorgruppe hinzugefügt/entfernt. Wenn Nodes zu einem Kubernetes-Cluster hinzugefügt werden, `trident-csi` DemonSet setzt einen POD ein (`trident-csi-xxxxx`) Auf den neu hinzugefügten Knoten und registriert die neuen Knoten kann es Volumes an. Node-IQNs werden ebenfalls zur iGroup des Backend hinzugefügt. Eine ähnliche Reihe von Schritten behandelt das Entfernen von IQNs, wenn Nodes aus Kubernetes abgesperrt, entleert und gelöscht werden.

Wenn Astra Trident nicht als CSI-Bereitstellung ausgeführt wird, muss die Initiatorgruppe manuell aktualisiert werden, um die iSCSI-IQNs von jedem Worker-Node im Kubernetes-Cluster zu enthalten. IQNs von Nodes, die dem Kubernetes-Cluster beitreten, müssen zur Initiatorgruppe hinzugefügt werden. Ebenso müssen IQNs von Nodes, die aus dem Kubernetes-Cluster entfernt werden, aus der Initiatorgruppe entfernt werden.

Verbindungen mit bidirektionalem CHAP authentifizieren

Astra Trident kann iSCSI-Sitzungen mit bidirektionalem CHAP für die authentifizieren `ontap-san` Und `ontap-san-economy` Treiber. Hierfür muss die Aktivierung von erforderlich sein `useCHAP` Option in der Back-End-Definition. Wenn eingestellt auf `true`, Astra Trident konfiguriert die Standard-Initiator-Sicherheit der SVM auf bidirektionales CHAP und legt den Benutzernamen und die Schlüssel aus der Backend-Datei. NetApp empfiehlt die Verwendung von bidirektionalem CHAP zur Authentifizierung von Verbindungen. Die folgende Beispielkonfiguration ist verfügbar:

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "FaKePaSsWoRd",
    "igroupName": "trident",
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}
```

Der `useCHAP` Parameter ist eine Boolesche Option, die nur einmal konfiguriert werden kann. Die Standardeinstellung ist „false“. Nachdem Sie die Einstellung auf „true“ gesetzt haben, können Sie sie nicht auf „false“ setzen.

Zusätzlich zu `useCHAP=true`, Das `chapInitiatorSecret`, `chapTargetInitiatorSecret`, `chapTargetUsername`, und `chapUsername` Felder müssen in die Backend-Definition aufgenommen werden. Die Geheimnisse können geändert werden, nachdem ein Backend durch Ausführen erstellt wird `tridentctl update`.

So funktioniert es

Nach Einstellung `useCHAP` Der Storage-Administrator weist Astra Trident an, CHAP im Storage-Back-End zu konfigurieren. Dazu gehört Folgendes:

- Einrichten von CHAP auf der SVM:
  - Wenn der Standardsicherheitstyp des SVM keine (standardmäßig eingestellt) ist **und** gibt es keine bereits vorhandenen LUNs im Volume, setzt Astra Trident den Standardsicherheitstyp auf `CHAP` Und fahren Sie mit der Konfiguration des CHAP-Initiators und des Zielbenutzernamens und der Schlüssel fort.
  - Wenn die SVM LUNs enthält, aktiviert Astra Trident nicht CHAP auf der SVM. Dadurch wird sichergestellt, dass der Zugriff auf LUNs, die bereits auf der SVM vorhanden sind, nicht beschränkt ist.
- Konfigurieren des CHAP-Initiators und des Ziel-Usernamens und der Schlüssel; diese Optionen müssen in der Back-End-Konfiguration angegeben werden (siehe oben).
- Verwaltung der Hinzufügung von Inititatoren zum `igroupName` Gegeben im Backend. Wenn die Angabe nicht festgelegt ist, wird standardmäßig auf diese Option gesetzt `trident`.

Nach der Erstellung des Backend erstellt Astra Trident eine entsprechende `tridentbackend` CRD: Speichert die CHAP-Geheimnisse und Benutzernamen als Kubernetes-Geheimnisse. Alle PVS, die von Astra Trident auf diesem Backend erstellt werden, werden über CHAP gemountet und angeschlossen.

Anmeldedaten rotieren und Back-Ends aktualisieren

Sie können die CHAP-Anmeldeinformationen aktualisieren, indem Sie die CHAP-Parameter im aktualisieren `backend.json` Datei: Dazu müssen die CHAP-Schlüssel aktualisiert und der verwendet werden `tridentctl update` Befehl zum Übergeben dieser Änderungen.

> Wenn Sie die CHAP-Schlüssel für ein Backend aktualisieren, müssen Sie verwenden `tridentctl` Um das Backend zu aktualisieren. Aktualisieren Sie die Anmeldeinformationen im Storage-Cluster nicht über die Benutzeroberfläche von CLI/ONTAP, da Astra Trident diese Änderungen nicht übernehmen kann.

```
$ cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "FaKePaSsWoRd",
    "igroupName": "trident",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

$ ./tridentctl update backend ontap_san_chap -f backend-san.json -n
trident
+----------------+----------------+--------------------------------------
+--------+---------+
|   NAME         | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+----------------+----------------+--------------------------------------
+--------+---------+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |       7 |
+----------------+----------------+--------------------------------------
+--------+---------+
```

Bestehende Verbindungen bleiben unbeeinträchtigt, sie bleiben auch weiterhin aktiv, wenn die Anmeldedaten vom Astra Trident auf der SVM aktualisiert werden. Neue Verbindungen verwenden die aktualisierten Anmeldedaten und vorhandene Verbindungen bleiben weiterhin aktiv. Wenn Sie alte PVS trennen und neu verbinden, werden sie die aktualisierten Anmeldedaten verwenden.

Konfigurationsoptionen und Beispiele

Erfahren Sie, wie Sie mit Ihrer Installation von Astra Trident ONTAP SAN-Treiber erstellen und verwenden. Dieser Abschnitt enthält Beispiele für die Back-End-Konfiguration und Details zur Zuordnung von Back-Ends zu StorageClasses.

Back-End-Konfigurationsoptionen

Die Back-End-Konfigurationsoptionen finden Sie in der folgenden Tabelle:

Parameter	Beschreibung	Standard
`version`		Immer 1
`storageDriverName`	Name des Speichertreibers	„ontap-nas“, „ontap-nas-Economy“, „ontap-nas-flexgroup“, „ontap-san“, „ontap-san-Economy“
`backendName`	Benutzerdefinierter Name oder das Storage-Backend	Treibername + „_“ + DatenLIF
`managementLIF`	IP-Adresse eines Clusters oder einer SVM-Management-LIF	„10.0.0.1“, „[2001:1234:abcd::fefe]“
`dataLIF`	IP-Adresse des LIF-Protokolls. Verwenden Sie eckige Klammern für IPv6. Kann nicht aktualisiert werden, nachdem Sie sie festgelegt haben	Abgeleitet von der SVM, sofern angegeben
`useCHAP`	Verwenden Sie CHAP, um iSCSI für ONTAP-SAN-Treiber zu authentifizieren [Boolesch]	Falsch
`chapInitiatorSecret`	CHAP-Initiatorschlüssel. Erforderlich, wenn `useCHAP=true`	„“
`labels`	Satz willkürlicher JSON-formatierter Etiketten für Volumes	„“
`chapTargetInitiatorSecret`	Schlüssel für CHAP-Zielinitiator. Erforderlich, wenn `useCHAP=true`	„“
`chapUsername`	Eingehender Benutzername. Erforderlich, wenn `useCHAP=true`	„“
`chapTargetUsername`	Zielbenutzername. Erforderlich, wenn `useCHAP=true`	„“
`clientCertificate`	Base64-codierter Wert des Clientzertifikats. Wird für zertifikatbasierte Authentifizierung verwendet	„“
`clientPrivateKey`	Base64-kodierte Wert des privaten Client-Schlüssels. Wird für zertifikatbasierte Authentifizierung verwendet	„“

Parameter	Beschreibung	Standard
`trustedCACertificate`	Base64-kodierte Wert des vertrauenswürdigen CA-Zertifikats. Optional Wird für zertifikatbasierte Authentifizierung verwendet	„“
`username`	Benutzername für die Verbindung mit dem Cluster/SVM. Wird für Anmeldeinformationsbasierte verwendet	„“
`password`	Passwort für die Verbindung mit dem Cluster/SVM Wird für Anmeldeinformationsbasierte verwendet	„“
`svm`	Zu verwendende Storage Virtual Machine	Abgeleitet wenn eine SVM `managementLIF` Angegeben ist
`igroupName`	Der Name der Initiatorgruppe für die zu verwendenden SAN Volumes	„Trident-<Backend-UUID>“
`storagePrefix`	Das Präfix wird beim Bereitstellen neuer Volumes in der SVM verwendet. Kann nicht aktualisiert werden, nachdem Sie sie festgelegt haben	„Dreizack“
`limitAggregateUsage`	Bereitstellung fehlgeschlagen, wenn die Nutzung über diesem Prozentsatz liegt. **Gilt nicht für Amazon FSX für ONTAP**	„“ (nicht standardmäßig durchgesetzt)
`limitVolumeSize`	Bereitstellung fehlgeschlagen, wenn die angeforderte Volume-Größe über diesem Wert liegt.	„“ (nicht standardmäßig durchgesetzt)
`lunsPerFlexvol`	Die maximale Anzahl an LUNs pro FlexVol muss im Bereich [50, 200] liegen.	„100“
`debugTraceFlags`	Fehler-Flags bei der Fehlerbehebung beheben. Beispiel: { „API“:false, „Methode“:true}	Null

Um mit dem ONTAP-Cluster zu kommunizieren, sollten Sie die Authentifizierungsparameter angeben. Dies kann der Benutzername/das Passwort für ein Sicherheitsanmeldung oder ein installiertes Zertifikat sein.

Wenn Sie ein Amazon FSX für das NetApp ONTAP-Backend verwenden, geben Sie das nicht an `limitAggregateUsage` Parameter. Der `fsxadmin` Und `vsadmin` Die von Amazon FSX für NetApp ONTAP bereitgestellten Rollen enthalten nicht die erforderlichen Zugriffsberechtigungen, um die Aggregatnutzung abzurufen und sie über Astra Trident zu begrenzen.

Verwenden Sie es nicht `debugTraceFlags` Es sei denn, Sie beheben Fehler und benötigen einen detaillierten Log Dump.

Für das `ontap-san` Treiber: Der Standard besteht darin, alle Daten-LIF-IPs der SVM zu verwenden und iSCSI Multipath zu verwenden. Angeben einer IP-Adresse für die Daten-LIF für das `ontap-san` Treiber zwingt sie, Multipath zu deaktivieren und nur die angegebene Adresse zu verwenden.

Denken Sie beim Erstellen eines Backend daran `dataLIF` Und `storagePrefix` Kann nach der Erstellung nicht geändert werden. Um diese Parameter zu aktualisieren, müssen Sie ein neues Backend erstellen.

`igroupName` Kann auf eine Initiatorgruppe festgelegt werden, die bereits auf dem ONTAP Cluster erstellt wurde. Wenn nicht angegeben, erstellt Astra Trident automatisch eine igroup mit dem Namen Trident-<Backend-UUID>. Bei Bereitstellung eines vordefinierten igroupName empfiehlt NetApp die Verwendung einer Initiatorgruppe pro Kubernetes Cluster, sofern die SVM zwischen Umgebungen gemeinsam genutzt werden soll. Dies ist für Astra Trident erforderlich, damit IQN-Ergänzungen/Löschungen automatisch durchgeführt werden können.

Bei Back-Ends können auch Initiatorgruppen nach der Erstellung aktualisiert werden:

- IgroupName kann aktualisiert werden, um auf eine neue Initiatorgruppe zu verweisen, die auf der SVM außerhalb des Astra Trident erstellt und gemanagt wird.
- Name der igroupName kann weggelassen werden. In diesem Fall erstellt und verwaltet Astra Trident automatisch eine Trident-<Backend-UUID> igroup.

In beiden Fällen können Sie weiterhin auf Volume-Anhänge zugreifen. Zukünftige Volume-Anhänge verwenden die aktualisierte Initiatorgruppe. Dieses Update wird den Zugriff auf Volumes im Backend nicht unterbrechen.

Für den kann ein vollständig qualifizierter Domänenname (FQDN) angegeben werden `managementLIF` Option.

```
`managementLIF` Für alle ONTAP-Treiber können auch IPv6-Adressen
eingestellt werden. Installieren Sie Trident zusammen mit dem `--use-ipv6`
Flagge. Es muss sorgfältig darauf achten, zu definieren `managementLIF`
IPv6-Adresse innerhalb von eckigen Klammern.
```

Stellen Sie beim Verwenden von IPv6-Adressen sicher `managementLIF` Und `dataLIF` (Falls in Ihrer Backend-Definition enthalten) sind innerhalb eckiger Klammern definiert, wie [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Wenn `dataLIF` Ist nicht angegeben, holt Astra Trident die IPv6 Daten-LIFs von der SVM ab.

Um die ontap-san-Treiber für die Verwendung von CHAP zu aktivieren, legen Sie den fest `useCHAP` Parameter an `true` Back-End-Definition: Astra Trident konfiguriert und verwendet dann bidirektionales CHAP als Standardauthentifizierung für die im Backend angegebene SVM. Siehe "Hier" Um zu erfahren, wie es funktioniert.

Für das `ontap-san-economy` Treiber, der `limitVolumeSize` Mit dieser Option wird auch die maximale Größe der Volumes eingeschränkt, die es für qtrees und LUNs verwaltet.

Astra Trident setzt Provisioning-Labels im Feld „Kommentare“ aller Volumes, die mit dem erstellt wurden `ontap-san` Treiber. Für jedes erstellte Volume wird das Feld „Kommentare“ auf der FlexVol mit allen Etiketten auf dem Speicherpool gefüllt, in dem es platziert wird. Storage-Administratoren können Labels pro Storage-Pool definieren und alle Volumes gruppieren, die in einem Storage-Pool erstellt wurden. Dies bietet eine praktische Möglichkeit, Volumes anhand einer Reihe anpassbarer Etiketten, die in der Backend-Konfiguration bereitgestellt werden, zu unterscheiden.

Back-End-Konfigurationsoptionen für die Bereitstellung von Volumes

Mit diesen Optionen kann standardmäßig gesteuert werden, wie jedes Volume in einem speziellen Abschnitt der Konfiguration bereitgestellt wird. Ein Beispiel finden Sie unten in den Konfigurationsbeispielen.

Parameter	Beschreibung	Standard
`spaceAllocation`	Speicherplatzzuweisung für LUNs	„Wahr“
`spaceReserve`	Space Reservation Mode; „none“ (Thin) oder „Volume“ (Thick)	„Keine“
`snapshotPolicy`	Die Snapshot-Richtlinie zu verwenden	„Keine“
`qosPolicy`	QoS-Richtliniengruppe zur Zuweisung für erstellte Volumes Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Storage Pool/Backend	„“
`adaptiveQosPolicy`	Adaptive QoS-Richtliniengruppe mit Zuordnung für erstellte Volumes Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Storage Pool/Backend	„“
`snapshotReserve`	Prozentsatz des für Snapshots reservierten Volumens „0“	Wenn `snapshotPolicy` Ist „keine“, sonst „“
`splitOnClone`	Teilen Sie einen Klon bei der Erstellung von seinem übergeordneten Objekt auf	„Falsch“
`splitOnClone`	Teilen Sie einen Klon bei der Erstellung von seinem übergeordneten Objekt auf	„Falsch“
`encryption`	NetApp Volume Encryption aktivieren	„Falsch“
`securityStyle`	Sicherheitstyp für neue Volumes	„unix“
`tieringPolicy`	Tiering-Richtlinie zur Verwendung von „keiner“	„Nur Snapshot“ für eine ONTAP 9.5 SVM-DR-Konfiguration

Die Verwendung von QoS Policy Groups mit Astra Trident erfordert ONTAP 9.8 oder höher. Es wird empfohlen, eine nicht gemeinsam genutzte QoS-Richtliniengruppe zu verwenden und sicherzustellen, dass die Richtliniengruppe auf jede Komponente einzeln angewendet wird. Eine Richtliniengruppe für Shared QoS führt zur Durchsetzung der Obergrenze für den Gesamtdurchsatz aller Workloads.

Hier ist ein Beispiel mit definierten Standardeinstellungen:

```
{
 "version": 1,
 "storageDriverName": "ontap-san",
 "managementLIF": "10.0.0.1",
 "dataLIF": "10.0.0.2",
 "svm": "trident_svm",
 "username": "admin",
 "password": "password",
 "labels": {"k8scluster": "dev2", "backend": "dev2-sanbackend"},
 "storagePrefix": "alternate-trident",
 "igroupName": "custom",
 "debugTraceFlags": {"api":false, "method":true},
 "defaults": {
     "spaceReserve": "volume",
     "qosPolicy": "standard",
     "spaceAllocation": "false",
     "snapshotPolicy": "default",
     "snapshotReserve": "10"
 }
}
```

Für alle mit dem erstellten Volumes `ontap-san` Treiber: Astra Trident fügt der FlexVol zusätzliche Kapazität von 10 % hinzu, um die LUN-Metadaten zu bewältigen. Die LUN wird genau mit der Größe bereitgestellt, die der Benutzer in der PVC anfordert. Astra Trident fügt 10 Prozent zum FlexVol hinzu (wird in ONTAP als verfügbare Größe dargestellt). Benutzer erhalten jetzt die Menge an nutzbarer Kapazität, die sie angefordert haben. Diese Änderung verhindert auch, dass LUNs schreibgeschützt werden, sofern der verfügbare Speicherplatz nicht vollständig genutzt wird. Dies gilt nicht für die Wirtschaft von ontap-san.

Für Back-Ends, die definieren `snapshotReserve`, Astra Trident berechnet die Größe der Volumes wie folgt:

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
percentage) / 100)] * 1.1
```

Das 1.1 ist der zusätzliche 10-Prozent-Astra Trident fügt dem FlexVol hinzu, um die LUN-Metadaten zu bewältigen. Für `snapshotReserve` = 5 %, und die PVC-Anforderung = 5 gib, die Gesamtgröße des Volumes beträgt 5,79 gib und die verfügbare Größe 5,5 gib. Der `volume show` Der Befehl sollte Ergebnisse anzeigen, die diesem Beispiel ähnlich sind:

Die Größenanpassung ist derzeit die einzige Möglichkeit, die neue Berechnung für ein vorhandenes Volume zu verwenden.

Minimale Konfigurationsbeispiele

Die folgenden Beispiele zeigen grundlegende Konfigurationen, bei denen die meisten Parameter standardmäßig belassen werden. Dies ist der einfachste Weg, ein Backend zu definieren.

> Wenn Sie Amazon FSX auf NetApp ONTAP mit Astra Trident verwenden, empfiehlt es sich, DNS-Namen für LIFs anstelle von IP-Adressen anzugeben.

`ontap-san` Treiber mit zertifikatbasierter Authentifizierung

Dies ist ein minimales Beispiel für die Back-End-Konfiguration. `clientCertificate`, `clientPrivateKey`, und `trustedCACertificate` (Optional, wenn Sie eine vertrauenswürdige CA verwenden) werden ausgefüllt `backend.json` Und nehmen Sie die base64-kodierten Werte des Clientzertifikats, des privaten Schlüssels und des vertrauenswürdigen CA-Zertifikats.

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "DefaultSANBackend",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
    "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
    "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz"
}
```

`ontap-san` Treiber mit bidirektionalem CHAP

Dies ist ein minimales Beispiel für die Back-End-Konfiguration. Mit dieser Grundkonfiguration wird ein erstellt

`ontap-san` Back-End mit `useCHAP` Auf einstellen `true`.

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "labels": {"k8scluster": "test-cluster-1", "backend": "testcluster1-
sanbackend"},
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "username": "vsadmin",
    "password": "secret"
}
```

`ontap-san-economy` **Treiber**

```
{
    "version": 1,
    "storageDriverName": "ontap-san-economy",
    "managementLIF": "10.0.0.1",
    "svm": "svm_iscsi_eco",
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "username": "vsadmin",
    "password": "secret"
}
```

Beispiele für Back-Ends mit virtuellen Storage-Pools

In der unten gezeigten Beispiel-Back-End-Definitionsdatei werden bestimmte Standardeinstellungen für alle Storage Pools festgelegt, z. B. `spaceReserve` Bei keiner, `spaceAllocation` Bei false, und `encryption` Bei false. Die virtuellen Speicherpools werden im Abschnitt Speicher definiert.

In diesem Beispiel legt ein Teil des Speicherpools seine eigenen fest `spaceReserve`, `spaceAllocation`, und `encryption` Werte und einige Pools überschreiben die oben festgelegten Standardwerte.

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "username": "vsadmin",
    "password": "secret",

    "defaults": {
          "spaceAllocation": "false",
          "encryption": "false",
          "qosPolicy": "standard"
    },
    "labels":{"store": "san_store", "kubernetes-cluster": "prod-cluster-
1"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"protection":"gold", "creditpoints":"40000"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceAllocation": "true",
                "encryption": "true",
                "adaptiveQosPolicy": "adaptive-extreme"
            }
        },
        {
            "labels":{"protection":"silver", "creditpoints":"20000"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceAllocation": "false",
                "encryption": "true",
                "qosPolicy": "premium"
            }
        },
        {
            "labels":{"protection":"bronze", "creditpoints":"5000"},
            "zone":"us_east_1c",
```

```
            "defaults": {
                "spaceAllocation": "true",
                "encryption": "false"
            }
        }
    ]
}
```

Hier ist ein iSCSI-Beispiel für das `ontap-san-economy` Treiber:

```
{
    "version": 1,
    "storageDriverName": "ontap-san-economy",
    "managementLIF": "10.0.0.1",
    "svm": "svm_iscsi_eco",
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "username": "vsadmin",
    "password": "secret",

    "defaults": {
          "spaceAllocation": "false",
          "encryption": "false"
    },
    "labels":{"store":"san_economy_store"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"app":"oracledb", "cost":"30"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceAllocation": "true",
                "encryption": "true"
            }
        },
        {
            "labels":{"app":"postgresdb", "cost":"20"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceAllocation": "false",
                "encryption": "true"
```

```
            }
        },
        {
            "labels":{"app":"mysqldb", "cost":"10"},
            "zone":"us_east_1c",
            "defaults": {
                "spaceAllocation": "true",
                "encryption": "false"
            }
        }
    ]
}
```

Back-Ends StorageClasses zuordnen

Die folgenden StorageClass-Definitionen beziehen sich auf die oben genannten virtuellen Speicherpools. Verwenden der `parameters.selector` Feld gibt in jeder StorageClass an, welche virtuellen Pools zum Hosten eines Volumes verwendet werden können. Auf dem Volume werden die Aspekte im ausgewählten virtuellen Pool definiert.

- Die erste StorageClass (`protection-gold`) Wird dem ersten, zweiten virtuellen Speicherpool in zugeordnet `ontap-nas-flexgroup` Back-End und der erste virtuelle Speicherpool im `ontap-san` Back-End: Dies sind die einzigen Pools, die Schutz auf Goldebene bieten.
- Die zweite StorageClass (`protection-not-gold`) Wird dem dritten, vierten virtuellen Speicherpool in zugeordnet `ontap-nas-flexgroup` Back-End und der zweite dritte virtuelle Speicherpool in `ontap-san` Back-End: Dies sind die einzigen Pools, die Schutz Level nicht Gold bieten.
- Die dritte StorageClass (`app-mysqldb`) Wird dem vierten virtuellen Speicherpool in zugeordnet `ontap-nas` Back-End und der dritte virtuelle Storage-Pool in `ontap-san-economy` Back-End: Dies sind die einzigen Pools, die eine Storage-Pool-Konfiguration für die mysqldb-Typ-App bieten.
- Die vierte StorageClass (`protection-silver-creditpoints-20k`) Wird dem dritten virtuellen Speicher-Pool in zugeordnet `ontap-nas-flexgroup` Back-End und der zweite virtuelle Storage-Pool in `ontap-san` Back-End: Dies sind die einzigen Pools, die Gold-Level-Schutz mit 20000 Kreditpunkten bieten.
- Die fünfte StorageClass (`creditpoints-5k`) Wird dem zweiten virtuellen Speicherpool in zugeordnet `ontap-nas-economy` Back-End und der dritte virtuelle Storage-Pool in `ontap-san` Back-End: Dies sind die einzigen Poolangebote mit 5000 Kreditpunkten.

Astra Trident entscheidet, welcher virtuelle Storage Pool ausgewählt wird und ob die Storage-Anforderungen erfüllt werden.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: netapp.io/trident
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Konfigurieren Sie ein Backend mit ONTAP-NAS-Treibern

Erfahren Sie mehr über die Konfiguration eines ONTAP-Backend mit ONTAP-NAS-Treibern.

- "Vorbereitung"
- "Konfiguration und Beispiele"

Benutzerberechtigungen

Astra Trident erwartet, dass er entweder als ONTAP- oder SVM-Administrator ausgeführt wird, in der Regel mit dem `admin` Cluster-Benutzer oder ein `vsadmin` SVM-Benutzer oder ein Benutzer mit einem anderen Namen und derselben Rolle. Astra Trident erwartet, dass bei Amazon FSX für Implementierungen von NetApp ONTAP, über das Cluster entweder als ONTAP- oder SVM-Administrator ausgeführt wird `fsxadmin` Benutzer oder A `vsadmin` SVM-Benutzer oder ein Benutzer mit einem anderen Namen und derselben Rolle. Der `fsxadmin` Der Benutzer ist ein eingeschränkter Ersatz für den Cluster-Admin-Benutzer.

> Wenn Sie den verwenden `limitAggregateUsage` Parameter, Berechtigungen für Cluster-Admin sind erforderlich. Bei der Verwendung von Amazon FSX für NetApp ONTAP mit Astra Trident, das `limitAggregateUsage` Der Parameter funktioniert nicht mit dem `vsadmin` Und `fsxadmin` Benutzerkonten. Der Konfigurationsvorgang schlägt fehl, wenn Sie diesen Parameter angeben.

Vorbereitung

Erfahren Sie, wie Sie ein ONTAP-Back-End mit ONTAP-NAS-Treibern vorbereiten. Für alle ONTAP Back-Ends benötigt Astra Trident mindestens ein Aggregat, das der SVM zugewiesen ist.

Für alle ONTAP Back-Ends benötigt Astra Trident mindestens ein Aggregat, das der SVM zugewiesen ist.

Denken Sie daran, dass Sie auch mehr als einen Treiber ausführen können und Speicherklassen erstellen können, die auf den einen oder anderen verweisen. Beispielsweise könnten Sie eine Gold-Klasse konfigurieren, die den verwendet `ontap-nas` Fahrer und eine Bronze-Klasse, die den verwendet `ontap-nas-economy` Eins.

Alle Kubernetes-Worker-Nodes müssen über die entsprechenden NFS-Tools verfügen. Siehe "Hier" Entnehmen.

Authentifizierung

Astra Trident bietet zwei Arten der Authentifizierung eines ONTAP-Backend.

- Anmeldeinformationsbasiert: Benutzername und Passwort für einen ONTAP-Benutzer mit den erforderlichen Berechtigungen. Es wird empfohlen, eine vordefinierte Sicherheits-Login-Rolle zu verwenden, wie z. B. `admin` Oder `vsadmin` Für maximale Kompatibilität mit ONTAP Versionen.
- Zertifikatsbasiert: Astra Trident kann auch mit einem ONTAP Cluster kommunizieren. Verwenden Sie dazu ein Zertifikat, das auf dem Backend installiert ist. Hier muss die Backend-Definition Base64-kodierte Werte des Client-Zertifikats, des Schlüssels und des vertrauenswürdigen CA-Zertifikats enthalten, sofern verwendet (empfohlen).

Benutzer können auch vorhandene Back-Ends aktualisieren, sich für die Migration von Anmeldeinformationsbasierten zu zertifikatbasierten Optionen entscheiden und umgekehrt. Wenn **sowohl Anmeldeinformationen als auch Zertifikate** bereitgestellt werden, verwendet Astra Trident standardmäßig Zertifikate, während eine Warnung ausgegeben wird, um die Anmeldeinformationen aus der Back-End-

Definition zu entfernen.

Aktivieren Sie die Anmeldeinformationsbasierte Authentifizierung

Astra Trident erfordert die Zugangsdaten für einen Administrator mit SVM-Umfang/Cluster-Umfang, um mit dem Backend von ONTAP zu kommunizieren. Es wird empfohlen, die Standard-vordefinierten Rollen wie zu verwenden `admin` Oder `vsadmin`. So ist gewährleistet, dass die Kompatibilität mit künftigen ONTAP Versionen gewährleistet ist, die FunktionsAPIs der künftigen Astra Trident Versionen bereitstellen können. Eine benutzerdefinierte Sicherheits-Login-Rolle kann mit Astra Trident erstellt und verwendet werden, wird aber nicht empfohlen.

Eine Beispiel-Back-End-Definition sieht folgendermaßen aus:

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "secret"
}
```

Beachten Sie, dass die Backend-Definition der einzige Ort ist, an dem die Anmeldeinformationen im reinen Text gespeichert werden. Nach der Erstellung des Backend werden Benutzernamen/Passwörter mit Base64 codiert und als Kubernetes Secrets gespeichert. Die Erstellung/Aktualisierung eines Backend ist der einzige Schritt, der Kenntnisse der Anmeldeinformationen erfordert. Daher ist dieser Vorgang nur für Administratoren und wird vom Kubernetes-/Storage-Administrator ausgeführt.

Aktivieren Sie die zertifikatbasierte Authentifizierung

Neue und vorhandene Back-Ends können ein Zertifikat verwenden und mit dem ONTAP-Back-End kommunizieren. In der Backend-Definition sind drei Parameter erforderlich.

- ClientCertificate: Base64-codierter Wert des Clientzertifikats.
- ClientPrivateKey: Base64-kodierte Wert des zugeordneten privaten Schlüssels.
- Trusted CACertificate: Base64-codierter Wert des vertrauenswürdigen CA-Zertifikats. Bei Verwendung einer vertrauenswürdigen CA muss dieser Parameter angegeben werden. Dies kann ignoriert werden, wenn keine vertrauenswürdige CA verwendet wird.

Ein typischer Workflow umfasst die folgenden Schritte.

Schritte

1. Erzeugen eines Clientzertifikats und eines Schlüssels. Legen Sie beim Generieren den allgemeinen Namen (CN) für den ONTAP-Benutzer fest, der sich authentifizieren soll als.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
```

2. Fügen Sie dem ONTAP-Cluster ein vertrauenswürdiges CA-Zertifikat hinzu. Dies kann möglicherweise bereits vom Storage-Administrator übernommen werden. Ignorieren, wenn keine vertrauenswürdige CA verwendet wird.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Installieren Sie das Client-Zertifikat und den Schlüssel (von Schritt 1) auf dem ONTAP-Cluster.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. Bestätigen Sie, dass die ONTAP-Sicherheitsanmeldungsrolle unterstützt wird `cert` Authentifizierungsmethode.

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

5. Testen Sie die Authentifizierung mithilfe des generierten Zertifikats. <ONTAP Management LIF> und <vServer Name> durch Management-LIF-IP und SVM-Namen ersetzen. Sie müssen sicherstellen, dass die Service-Richtlinie für das LIF auf festgelegt ist `default-data-management`.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Encodieren von Zertifikat, Schlüssel und vertrauenswürdigem CA-Zertifikat mit Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Erstellen Sie das Backend mit den Werten, die aus dem vorherigen Schritt ermittelt wurden.

```
$ cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
$ tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Aktualisieren Sie Authentifizierungsmethoden, oder drehen Sie die Anmeldedaten

Sie können ein vorhandenes Backend aktualisieren, um eine andere Authentifizierungsmethode zu verwenden oder um ihre Anmeldeinformationen zu drehen. Das funktioniert auf beide Arten: Back-Ends, die einen Benutzernamen/ein Passwort verwenden, können aktualisiert werden, um Zertifikate zu verwenden; Back-Ends, die Zertifikate verwenden, können auf Benutzername/Passwort-basiert aktualisiert werden. Verwenden Sie dazu ein aktualisiertes `backend.json` Datei mit den erforderlichen Parametern für die Ausführung `tridentctl backend update`.

```
$ cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "secret",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
$ tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Bei der Änderung von Passwörtern muss der Speicheradministrator das Kennwort für den Benutzer auf ONTAP aktualisieren. Auf diese Weise folgt ein Backend-Update. Beim Drehen von Zertifikaten können dem Benutzer mehrere Zertifikate hinzugefügt werden. Das Backend wird dann aktualisiert und verwendet das neue Zertifikat. Danach kann das alte Zertifikat aus dem ONTAP Cluster gelöscht werden.

Durch die Aktualisierung eines Backend wird der Zugriff auf Volumes, die bereits erstellt wurden, nicht unterbrochen, und auch die danach erstellten Volume-Verbindungen werden beeinträchtigt. Ein erfolgreiches Backend-Update zeigt, dass Astra Trident mit dem ONTAP-Backend kommunizieren und zukünftige Volume-Operationen verarbeiten kann.

Management der NFS-Exportrichtlinien

Astra Trident verwendet NFS-Exportrichtlinien, um den Zugriff auf die Volumes zu kontrollieren, die er bereitstellt.

Astra Trident bietet zwei Optionen für die Arbeit mit Exportrichtlinien:

- Astra Trident kann die Exportrichtlinie selbst dynamisch managen. In diesem Betriebsmodus spezifiziert der Storage-Administrator eine Liste mit CIDR-Blöcken, die zulässige IP-Adressen darstellen. Astra Trident fügt automatisch Node-IPs hinzu, die in diese Bereiche fallen, zur Exportrichtlinie hinzu. Wenn keine

CIDRs angegeben werden, wird alternativ jede auf den Knoten gefundene globale Unicast-IP mit globalem Umfang zur Exportrichtlinie hinzugefügt.

- Storage-Administratoren können eine Exportrichtlinie erstellen und Regeln manuell hinzufügen. Astra Trident verwendet die Standard-Exportrichtlinie, es sei denn, in der Konfiguration ist ein anderer Name der Exportrichtlinie angegeben.

Dynamisches Managen von Exportrichtlinien

Mit der Version 20.04 von CSI Trident können Exportrichtlinien für ONTAP-Back-Ends dynamisch gemanagt werden. So kann der Storage-Administrator einen zulässigen Adressraum für Worker-Node-IPs festlegen, anstatt explizite Regeln manuell zu definieren. Dies vereinfacht das Management von Exportrichtlinien erheblich. Änderungen der Exportrichtlinie erfordern keine manuellen Eingriffe des Storage-Clusters mehr. Darüber hinaus hilft dies, den Zugriff auf das Storage-Cluster nur auf Worker-Nodes zu beschränken, die IPs im angegebenen Bereich besitzen und ein fein geregtes und automatisiertes Management unterstützen.

Das dynamische Management der Exportrichtlinien steht nur für CSI Trident zur Verfügung. Es ist wichtig sicherzustellen, dass die Worker Nodes nicht NATed werden.

Beispiel

Es müssen zwei Konfigurationsoptionen verwendet werden. Hier ist ein Beispiel Backend Definition:

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "ontap_nas_auto_export,
    "managementLIF": "192.168.0.135",
    "svm": "svm1",
    "username": "vsadmin",
    "password": "FaKePaSsWoRd",
    "autoExportCIDRs": ["192.168.0.0/24"],
    "autoExportPolicy": true
}
```

Bei Verwendung dieser Funktion müssen Sie sicherstellen, dass für die Root-Verbindung in Ihrer SVM eine vorab erstellte Exportrichtlinie mit einer Exportregel zur Verfügung steht, die den CIDR-Block des Nodes zulässt (z. B. die standardmäßige Exportrichtlinie). Folgen Sie immer der von NetApp empfohlenen Best Practice, eine SVM für Astra Trident einzurichten.

Hier ist eine Erklärung, wie diese Funktion funktioniert, anhand des obigen Beispiels:

- `autoExportPolicy` Ist auf festgelegt `true`. Dies zeigt an, dass Astra Trident eine Exportrichtlinie für den erstellen wird `svm1` SVM und das Hinzufügen und Löschen von Regeln mit behandeln `autoExportCIDRs` Adressblöcke. Beispiel: Ein Backend mit UUID 403b5326-8482-40db-96d0-d83fb3f4daec und `autoExportPolicy` Auf einstellen `true` Erstellt eine Exportrichtlinie mit dem Namen `trident-403b5326-8482-40db-96d0-d83fb3f4daec` Auf der SVM.
- `autoExportCIDRs` Enthält eine Liste von Adressblöcken. Dieses Feld ist optional und standardmäßig [„0.0.0.0/0“, „:/0“]. Falls nicht definiert, fügt Astra Trident alle Unicast-Adressen mit globellem Umfang hinzu, die auf den Worker-Nodes gefunden wurden.

In diesem Beispiel ist der `192.168.0.0/24` Adressbereich wird bereitgestellt. Das zeigt an, dass die Kubernetes-Node-IPs, die in diesen Adressbereich fallen, der vom Astra Trident erstellten Exportrichtlinie hinzugefügt werden. Wenn Astra Trident einen Knoten registriert, auf dem er ausgeführt wird, ruft er die IP-Adressen des Knotens ab und überprüft sie auf die in angegebenen Adressblöcke `autoExportCIDRs`. Nach dem Filtern der IPs erstellt Astra Trident Regeln für die Exportrichtlinie für die erkannte Client-IPs. Dabei gilt für jeden Node eine Regel, die er identifiziert.

Sie können aktualisieren `autoExportPolicy` Und `autoExportCIDRs` Für Back-Ends, nachdem Sie sie erstellt haben. Sie können neue CIDRs für ein Backend anhängen, das automatisch verwaltet wird oder vorhandene CIDRs löschen. Beim Löschen von CIDRs Vorsicht walten lassen, um sicherzustellen, dass vorhandene Verbindungen nicht unterbrochen werden. Sie können auch wählen, zu deaktivieren `autoExportPolicy` Für ein Backend und kehren Sie zu einer manuell erstellten Exportrichtlinie zurück. Dazu muss die Einstellung festgelegt werden `exportPolicy` Parameter in Ihrer Backend-Konfiguration.

Nachdem Astra Trident ein Backend erstellt oder aktualisiert hat, können Sie das Backend mit überprüfen `tridentctl` Oder das entsprechende `tridentbackend` CRD:

```
$ ./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileSystemType: ext4
```

Wenn Nodes zu einem Kubernetes-Cluster hinzugefügt und beim Astra Trident Controller registriert werden, werden die Exportrichtlinien vorhandener Back-Ends aktualisiert (vorausgesetzt, sie sind in den in angegebenen Adressbereich enthalten `autoExportCIDRs` Für das Backend).

Wenn ein Node entfernt wird, überprüft Astra Trident alle Back-Ends, die online sind, um die Zugriffsregel für den Node zu entfernen. Indem Astra Trident diese Node-IP aus den Exportrichtlinien für gemanagte Back-Ends entfernt, verhindert er abnormale Mounts, sofern diese IP nicht von einem neuen Node im Cluster verwendet wird.

Aktualisieren Sie bei zuvor vorhandenen Back-Ends das Backend mit `tridentctl update backend` Stellt sicher, dass Astra Trident die Exportrichtlinien automatisch verwaltet. Dadurch wird eine neue Exportrichtlinie

erstellt, die nach der UUID des Backend benannt ist und Volumes, die auf dem Backend vorhanden sind, verwenden die neu erstellte Exportrichtlinie, wenn sie erneut gemountet werden.

> Wenn Sie ein Backend mit automatisch gemanagten Exportrichtlinien löschen, wird die dynamisch erstellte Exportrichtlinie gelöscht. Wenn das Backend neu erstellt wird, wird es als neues Backend behandelt und erzeugt eine neue Exportrichtlinie.

Wenn die IP-Adresse eines aktiven Node aktualisiert wird, müssen Sie den Astra Trident Pod auf dem Node neu starten. Astra Trident aktualisiert dann die Exportrichtlinie für Back-Ends, die es verwaltet, um diese IP-Änderung zu berücksichtigen.

Konfigurationsoptionen und Beispiele

Erfahren Sie, wie Sie mit Ihrer Installation von Astra Trident ONTAP NAS-Treiber erstellen und verwenden. Dieser Abschnitt enthält Beispiele für die Back-End-Konfiguration und Details zur Zuordnung von Back-Ends zu StorageClasses.

Back-End-Konfigurationsoptionen

Die Back-End-Konfigurationsoptionen finden Sie in der folgenden Tabelle:

Parameter	Beschreibung	Standard
`version`		Immer 1
`storageDriverName`	Name des Speichertreibers	„ontap-nas“, „ontap-nas-Economy“, „ontap-nas-flexgroup“, „ontap-san“, „ontap-san-Economy“
`backendName`	Benutzerdefinierter Name oder das Storage-Backend	Treibername + „_“ + DatenLIF
`managementLIF`	IP-Adresse eines Clusters oder einer SVM-Management-LIF	„10.0.0.1“, „[2001:1234:abcd::fefe]“
`dataLIF`	IP-Adresse des LIF-Protokolls. Verwenden Sie eckige Klammern für IPv6. Kann nicht aktualisiert werden, nachdem Sie sie festgelegt haben	Abgeleitet von der SVM, sofern angegeben
`autoExportPolicy`	Automatische Erstellung von Exportrichtlinien aktivieren und [Boolean] aktualisieren	Falsch
`autoExportCIDRs`	Liste der CIDRs, um die Kubernetes-Knoten-IPs gegen Wann zu filtern `autoExportPolicy` Ist aktiviert	[„0.0.0.0/0“, „:/0“]`
`labels`	Satz willkürlicher JSON-formatierter Etiketten für Volumes	„“
`clientCertificate`	Base64-codierter Wert des Clientzertifikats. Wird für zertifikatbasierte Authentifizierung verwendet	„“

Parameter	Beschreibung	Standard
`clientPrivateKey`	Base64-kodierte Wert des privaten Client-Schlüssels. Wird für zertifikatbasierte Authentifizierung verwendet	„“
`trustedCACertificate`	Base64-kodierte Wert des vertrauenswürdigen CA-Zertifikats. Optional Wird für zertifikatbasierte Authentifizierung verwendet	„“
`username`	Benutzername für die Verbindung mit dem Cluster/SVM. Wird für Anmeldeinformationsbasierte verwendet	
`password`	Passwort für die Verbindung mit dem Cluster/SVM Wird für Anmeldeinformationsbasierte verwendet	
`svm`	Zu verwendende Storage Virtual Machine	Abgeleitet wenn eine SVM `managementLIF` Angegeben ist
`igroupName`	Der Name der Initiatorgruppe für die zu verwendenden SAN Volumes	„Trident-<Backend-UUID>“
`storagePrefix`	Das Präfix wird beim Bereitstellen neuer Volumes in der SVM verwendet. Kann nicht aktualisiert werden, nachdem Sie sie festgelegt haben	„Dreizack“
`limitAggregateUsage`	Bereitstellung fehlgeschlagen, wenn die Nutzung über diesem Prozentsatz liegt. **Gilt nicht für Amazon FSX für ONTAP**	„“ (nicht standardmäßig durchgesetzt)
`limitVolumeSize`	Bereitstellung fehlgeschlagen, wenn die angeforderte Volume-Größe über diesem Wert liegt.	„“ (nicht standardmäßig durchgesetzt)
`lunsPerFlexvol`	Die maximale Anzahl an LUNs pro FlexVol muss im Bereich [50, 200] liegen.	„100“
`debugTraceFlags`	Fehler-Flags bei der Fehlerbehebung beheben. Beispiel: { „API“:false, „Methode“:true}	Null
`nfsMountOptions`	Kommagetrennte Liste von NFS-Mount-Optionen	„“
`qtreesPerFlexvol`	Maximale Ques pro FlexVol, muss im Bereich [50, 300] liegen	„200“

Parameter	Beschreibung	Standard
`useREST`	Boolescher Parameter zur Verwendung von ONTAP REST-APIs. **Technische Vorschau**	Falsch

`useREST` Wird als **Tech-Vorschau bereitgestellt**, das für Testumgebungen und nicht für Produktions-Workloads empfohlen wird. Wenn eingestellt auf `true`, Astra Trident wird ONTAP REST APIs zur Kommunikation mit dem Backend verwenden. Diese Funktion erfordert ONTAP 9.8 und höher. Darüber hinaus muss die verwendete ONTAP-Login-Rolle Zugriff auf den haben `ontap` Applikation. Dies wird durch die vordefinierte zufrieden `vsadmin` Und `cluster-admin` Rollen:

Um mit dem ONTAP-Cluster zu kommunizieren, sollten Sie die Authentifizierungsparameter angeben. Dies kann der Benutzername/das Passwort für ein Sicherheitsanmeldung oder ein installiertes Zertifikat sein.

Wenn Sie ein Amazon FSX für das NetApp ONTAP-Backend verwenden, geben Sie das nicht an `limitAggregateUsage` Parameter. Der `fsxadmin` Und `vsadmin` Die von Amazon FSX für NetApp ONTAP bereitgestellten Rollen enthalten nicht die erforderlichen Zugriffsberechtigungen, um die Aggregatnutzung abzurufen und sie über Astra Trident zu begrenzen.

Verwenden Sie es nicht `debugTraceFlags` Es sei denn, Sie beheben Fehler und benötigen einen detaillierten Log Dump.

Denken Sie beim Erstellen eines Backend daran, dass das `dataLIF` Und `storagePrefix` Kann nach der Erstellung nicht geändert werden. Um diese Parameter zu aktualisieren, müssen Sie ein neues Backend erstellen.

Für den kann ein vollständig qualifizierter Domänenname (FQDN) angegeben werden `managementLIF` Option. Ein FQDN kann auch für den angegeben werden `dataLIF` Option, in diesem Fall wird der FQDN für die NFS-Mount-Vorgänge verwendet. Auf diese Weise können Sie ein Round Robin-DNS für den Lastausgleich über mehrere Daten-LIFs hinweg erstellen.

```
`managementLIF` Für alle ONTAP-Treiber können auch IPv6-Adressen
eingestellt werden. Installieren Sie unbedingt Astra Trident mit dem `--
use-ipv6` Flagge. Es ist darauf zu achten, das zu definieren
`managementLIF` IPv6-Adresse innerhalb von eckigen Klammern.
```

Stellen Sie beim Verwenden von IPv6-Adressen sicher `managementLIF` Und `dataLIF` (Falls in Ihrer Backend-Definition enthalten) sind innerhalb eckiger Klammern definiert, wie [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Wenn `dataLIF` Ist nicht angegeben, holt Astra Trident die IPv6 Daten-LIFs von der SVM ab.

Verwenden der `autoExportPolicy` Und `autoExportCIDRs` Optionen: CSI Trident kann Exportrichtlinien automatisch verwalten. Dies wird für alle ontap-nas-* Treiber unterstützt.

Für das `ontap-nas-economy` Treiber, der `limitVolumeSize` Die Option beschränkt auch die maximale Größe der Volumes, die es für qtrees und LUNs verwaltet, sowie die `qtreesPerFlexvol` Mit Option kann die

maximale Anzahl von qtrees pro FlexVol angepasst werden.

Der `nfsMountOptions` Parameter kann verwendet werden, um Mount-Optionen festzulegen. Die Mount-Optionen für persistente Kubernetes-Volumes werden normalerweise in Storage-Klassen angegeben. Wenn jedoch keine Mount-Optionen in einer Storage-Klasse angegeben sind, wird Astra Trident zu den Mount-Optionen zurückkehren, die in der Konfigurationsdatei des Storage-Back-End angegeben sind. Wenn in der Storage-Klasse oder der Konfigurationsdatei keine Mount-Optionen angegeben sind, setzt Astra Trident keine Mount-Optionen für ein damit verbundener persistenter Volume ein.

> Astra Trident setzt Provisioning-Labels im Feld „Kommentare“ aller Volumes, die mit erstellt wurden(`ontap-nas` Und(`ontap-nas-flexgroup`. Basierend auf dem verwendeten Treiber werden die Kommentare auf dem FlexVol festgelegt (`ontap-nas`) Oder FlexGroup (`ontap-nas-flexgroup`). Astra Trident kopiert zum Zeitpunkt der Bereitstellung alle auf einem Storage-Pool vorhandenen Labels auf das Storage-Volume. Storage-Administratoren können Labels pro Storage-Pool definieren und alle Volumes gruppieren, die in einem Storage-Pool erstellt wurden. Dies bietet eine praktische Möglichkeit, Volumes anhand einer Reihe anpassbarer Etiketten, die in der Backend-Konfiguration bereitgestellt werden, zu unterscheiden.

Back-End-Konfigurationsoptionen für die Bereitstellung von Volumes

Mit diesen Optionen kann standardmäßig gesteuert werden, wie jedes Volume in einem speziellen Abschnitt der Konfiguration bereitgestellt wird. Ein Beispiel finden Sie unten in den Konfigurationsbeispielen.

Parameter	Beschreibung	Standard
`spaceAllocation`	Speicherplatzzuweisung für LUNs	„Wahr“
`spaceReserve`	Space Reservation Mode; „none“ (Thin) oder „Volume“ (Thick)	„Keine“
`snapshotPolicy`	Die Snapshot-Richtlinie zu verwenden	„Keine“
`qosPolicy`	QoS-Richtliniengruppe zur Zuweisung für erstellte Volumes Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Storage Pool/Backend	„“
`adaptiveQosPolicy`	Adaptive QoS-Richtliniengruppe mit Zuordnung für erstellte Volumes Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Storage Pool/Backend. Nicht unterstützt durch ontap-nas-Ökonomie	„“
`snapshotReserve`	Prozentsatz des für Snapshots reservierten Volumens „0“	Wenn `snapshotPolicy` Ist „keine“, sonst „“
`splitOnClone`	Teilen Sie einen Klon bei der Erstellung von seinem übergeordneten Objekt auf	„Falsch“
`encryption`	NetApp Volume Encryption aktivieren	„Falsch“

Parameter	Beschreibung	Standard
`securityStyle`	Sicherheitstyp für neue Volumes	„unix“
`tieringPolicy`	Tiering-Richtlinie zur Verwendung von „keiner“	„Nur Snapshot“ für eine ONTAP 9.5 SVM-DR-Konfiguration
UnxPermissions	Modus für neue Volumes	„777“
Snapshots	Steuert die Sichtbarkeit des `.snapshot` Verzeichnis	„Falsch“
Exportpolitik	Zu verwendende Exportrichtlinie	„Standard“
Sicherheitstyp	Sicherheitstyp für neue Volumes	„unix“

Die Verwendung von QoS Policy Groups mit Astra Trident erfordert ONTAP 9.8 oder höher. Es wird empfohlen, eine nicht gemeinsam genutzte QoS-Richtliniengruppe zu verwenden und sicherzustellen, dass die Richtliniengruppe auf jede Komponente einzeln angewendet wird. Eine Richtliniengruppe für Shared QoS führt zur Durchsetzung der Obergrenze für den Gesamtdurchsatz aller Workloads.

Hier ist ein Beispiel mit definierten Standardeinstellungen:

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "customBackendName",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "labels": {"k8scluster": "dev1", "backend": "dev1-nasbackend"},
  "svm": "trident_svm",
  "username": "cluster-admin",
  "password": "password",
  "limitAggregateUsage": "80%",
  "limitVolumeSize": "50Gi",
  "nfsMountOptions": "nfsvers=4",
  "debugTraceFlags": {"api":false, "method":true},
  "defaults": {
    "spaceReserve": "volume",
    "qosPolicy": "premium",
    "exportPolicy": "myk8scluster",
    "snapshotPolicy": "default",
    "snapshotReserve": "10"
  }
}
```

Für `ontap-nas` Und `ontap-nas-flexgroups`Astra Trident verwendet jetzt eine neue Berechnung, um sicherzustellen, dass die FlexVol korrekt mit dem Prozentwert der Snapshot Reserve und PVC dimensioniert ist. Wenn der Benutzer eine PVC anfordert,

erstellt Astra Trident unter Verwendung der neuen Berechnung die ursprüngliche FlexVol mit mehr Speicherplatz. Diese Berechnung stellt sicher, dass der Benutzer den beschreibbaren Speicherplatz erhält, für den er in der PVC benötigt wird, und nicht weniger Speicherplatz als der angeforderte. Vor Version 2.07, wenn der Benutzer eine PVC anfordert (z. B. 5 gib), bei der SnapshotReserve auf 50 Prozent, erhalten sie nur 2,5 gib schreibbaren Speicherplatz. Der Grund dafür ist, dass der Benutzer das gesamte Volume und angefordert hat `snapshotReserve` Ist ein Prozentsatz davon. Mit Trident 21.07 sind die Benutzeranforderungen der beschreibbare Speicherplatz, und Astra Trident definiert den `snapshotReserve` Zahl als Prozentsatz des gesamten Volumens. Dies gilt nicht für `ontap-nas-economy`. Im folgenden Beispiel sehen Sie, wie das funktioniert:

Die Berechnung ist wie folgt:

```
Total volume size = (PVC requested size) / (1 - (snapshotReserve
percentage) / 100)
```

Für die snapshotReserve = 50 %, und die PVC-Anfrage = 5 gib, beträgt die Gesamtgröße des Volumes 2/.5 = 10 gib, und die verfügbare Größe beträgt 5 gib. Dies entspricht dem, was der Benutzer in der PVC-Anfrage angefordert hat. Der `volume show` Der Befehl sollte Ergebnisse anzeigen, die diesem Beispiel ähnlich sind:

Vorhandene Back-Ends aus vorherigen Installationen stellen Volumes wie oben beschrieben beim Upgrade von Astra Trident bereit. Bei Volumes, die Sie vor dem Upgrade erstellt haben, sollten Sie die Größe ihrer Volumes entsprechend der zu beobachtenden Änderung anpassen. Beispiel: Ein 2 gib PVC mit `snapshotReserve=50` Früher hat ein Volume ergeben, das 1 gib beschreibbaren Speicherplatz bereitstellt. Wenn Sie die Größe des Volumes auf 3 gib ändern, z. B. stellt die Applikation auf einem 6 gib an beschreibbarem Speicherplatz bereit.

Minimale Konfigurationsbeispiele

Die folgenden Beispiele zeigen grundlegende Konfigurationen, bei denen die meisten Parameter standardmäßig belassen werden. Dies ist der einfachste Weg, ein Backend zu definieren.

> Wenn Sie Amazon FSX auf NetApp ONTAP mit Trident verwenden, empfiehlt es sich, DNS-Namen für LIFs anstelle von IP-Adressen anzugeben.

`ontap-nas` Treiber mit zertifikatbasierter Authentifizierung

Dies ist ein minimales Beispiel für die Back-End-Konfiguration. `clientCertificate`, `clientPrivateKey`, und `trustedCACertificate` (Optional, wenn Sie eine vertrauenswürdige CA verwenden) werden ausgefüllt `backend.json` Und nehmen Sie die base64-kodierten Werte des Clientzertifikats, des privaten Schlüssels und des vertrauenswürdigen CA-Zertifikats.

```
{
  "version": 1,
  "backendName": "DefaultNASBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.15",
  "svm": "nfs_svm",
  "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
  "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
  "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz",
  "storagePrefix": "myPrefix_"
}
```

`ontap-nas` Treiber mit automatischer Exportrichtlinie

In diesem Beispiel erfahren Sie, wie Sie Astra Trident anweisen können, dynamische Exportrichtlinien zu verwenden, um die Exportrichtlinie automatisch zu erstellen und zu verwalten. Das funktioniert auch für das `ontap-nas-economy` Und `ontap-nas-flexgroup` Treiber.

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "labels": {"k8scluster": "test-cluster-east-1a", "backend": "test1-
nasbackend"},
    "autoExportPolicy": true,
    "autoExportCIDRs": ["10.0.0.0/24"],
    "username": "admin",
    "password": "secret",
    "nfsMountOptions": "nfsvers=4",
}
```

`ontap-nas-flexgroup` Treiber

{ „Version“: 1, „storageDriverName“: „ontap-nas-Flexgroup“, „ManagementLIF“: „10.0.0.1“, „dataLIF“: „10.0.0.2“, „Labels“: {„k8scluster“: „Test-Cluster-East-1b“, „Backend“: „test1-ontap-Cluster“}, „svm“: „svm_nfs“, „Benutzername“: „Vsadmin“, „Passwort“: „Secret“, }

`ontap-nas` Treiber mit IPv6

```
{
 "version": 1,
 "storageDriverName": "ontap-nas",
 "backendName": "nas_ipv6_backend",
 "managementLIF": "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]",
 "labels": {"k8scluster": "test-cluster-east-1a", "backend": "test1-ontap-
ipv6"},
 "svm": "nas_ipv6_svm",
 "username": "vsadmin",
 "password": "netapp123"
}
```

`ontap-nas-economy` **Treiber**

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret"
}
```

Beispiele für Back-Ends mit virtuellen Storage-Pools

In der unten gezeigten Beispiel-Back-End-Definitionsdatei werden bestimmte Standardeinstellungen für alle Storage Pools festgelegt, z. B. `spaceReserve` Bei keiner, `spaceAllocation` Bei false, und `encryption` Bei false. Die virtuellen Speicherpools werden im Abschnitt Speicher definiert.

In diesem Beispiel legt ein Teil des Speicherpools seine eigenen fest `spaceReserve`, `spaceAllocation`, und `encryption` Werte und einige Pools überschreiben die oben festgelegten Standardwerte.

`ontap-nas` **Treiber**

```
{
    {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "admin",
    "password": "secret",
```

```
    "nfsMountOptions": "nfsvers=4",

    "defaults": {
          "spaceReserve": "none",
          "encryption": "false",
          "qosPolicy": "standard"
    },
    "labels":{"store":"nas_store", "k8scluster": "prod-cluster-1"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"app":"msoffice", "cost":"100"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755",
                "adaptiveQosPolicy": "adaptive-premium"
            }
        },
        {
            "labels":{"app":"slack", "cost":"75"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"app":"wordpress", "cost":"50"},
            "zone":"us_east_1c",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0775"
            }
        },
        {
            "labels":{"app":"mysqldb", "cost":"25"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
```

```
        }
    ]
}
```

`ontap-nas-flexgroup` **Treiber**

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",

    "defaults": {
          "spaceReserve": "none",
          "encryption": "false"
    },
    "labels":{"store":"flexgroup_store", "k8scluster": "prod-cluster-1"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"protection":"gold", "creditpoints":"50000"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"protection":"gold", "creditpoints":"30000"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"protection":"silver", "creditpoints":"20000"},
            "zone":"us_east_1c",
            "defaults": {
                "spaceReserve": "none",
```

```
                "encryption": "true",
                "unixPermissions": "0775"
            }
        },
        {
            "labels":{"protection":"bronze", "creditpoints":"10000"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
        }
    ]
}
```

`ontap-nas-economy` **Treiber**

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",

    "defaults": {
          "spaceReserve": "none",
          "encryption": "false"
    },
    "labels":{"store":"nas_economy_store"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"department":"finance", "creditpoints":"6000"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"department":"legal", "creditpoints":"5000"},
```

```
            "zone":"us_east_1b",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"department":"engineering", "creditpoints":"3000"},
            "zone":"us_east_1c",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0775"
            }
        },
        {
            "labels":{"department":"humanresource",
"creditpoints":"2000"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
        }
    ]
}
```

Back-Ends StorageClasses zuordnen

Die folgenden StorageClass-Definitionen beziehen sich auf die oben genannten virtuellen Speicherpools. Verwenden der `parameters.selector` Feld gibt in jeder StorageClass an, welche virtuellen Pools zum Hosten eines Volumes verwendet werden können. Auf dem Volume werden die Aspekte im ausgewählten virtuellen Pool definiert.

- Die erste StorageClass (`protection-gold`) Wird dem ersten, zweiten virtuellen Speicherpool in zugeordnet `ontap-nas-flexgroup` Back-End und der erste virtuelle Speicherpool im `ontap-san` Back-End: Dies sind die einzigen Pools, die Schutz auf Goldebene bieten.
- Die zweite StorageClass (`protection-not-gold`) Wird dem dritten, vierten virtuellen Speicherpool in zugeordnet `ontap-nas-flexgroup` Back-End und der zweite dritte virtuelle Speicherpool in `ontap-san` Back-End: Dies sind die einzigen Pools, die Schutz Level nicht Gold bieten.
- Die dritte StorageClass (`app-mysqldb`) Wird dem vierten virtuellen Speicherpool in zugeordnet `ontap-nas` Back-End und der dritte virtuelle Storage-Pool in `ontap-san-economy` Back-End: Dies sind die einzigen Pools, die eine Storage-Pool-Konfiguration für die mysqldb-Typ-App bieten.
- Die vierte StorageClass (`protection-silver-creditpoints-20k`) Wird dem dritten virtuellen Speicher-Pool in zugeordnet `ontap-nas-flexgroup` Back-End und der zweite virtuelle Storage-Pool in

`ontap-san` Back-End: Dies sind die einzigen Pools, die Gold-Level-Schutz mit 20000 Kreditpunkten bieten.

- Die fünfte StorageClass (`creditpoints-5k`) Wird dem zweiten virtuellen Speicherpool in zugeordnet `ontap-nas-economy` Back-End und der dritte virtuelle Storage-Pool in `ontap-san` Back-End: Dies sind die einzigen Poolangebote mit 5000 Kreditpunkten.

Astra Trident entscheidet, welcher virtuelle Storage Pool ausgewählt wird und ob die Storage-Anforderungen erfüllt werden.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: netapp.io/trident
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Setzen Sie Astra Trident mit Amazon FSX für NetApp ONTAP ein

"Amazon FSX für NetApp ONTAP", Ist ein vollständig gemanagter AWS Service, mit dem Kunden Filesysteme auf Basis des NetApp ONTAP Storage-Betriebssystems starten und ausführen können. Mit Amazon FSX für NetApp ONTAP können Sie bereits bekannte NetApp Funktionen sowie die Performance und Administration nutzen und gleichzeitig die Einfachheit, Agilität, Sicherheit und Skalierbarkeit beim Speichern von Daten in AWS nutzen. FSX unterstützt viele der ONTAP Dateisystemfunktionen und Administrations-APIs.

Ein Dateisystem ist die primäre Ressource in Amazon FSX, analog zu einem ONTAP-Cluster vor Ort. Innerhalb jeder SVM können Sie ein oder mehrere Volumes erstellen, bei denen es sich um Daten-Container handelt, die die Dateien und Ordner im Filesystem speichern. Amazon FSX für NetApp ONTAP wird Data ONTAP als gemanagtes Dateisystem in der Cloud zur Verfügung stellen. Der neue Dateisystemtyp heißt **NetApp ONTAP**.

Durch den Einsatz von Astra Trident mit Amazon FSX für NetApp ONTAP können Sie sicherstellen, dass die Kubernetes Cluster, die im Amazon Elastic Kubernetes Service (EKS) ausgeführt werden, persistente Block- und Datei-Volumes bereitstellen können, die von ONTAP unterstützt werden.

Erfahren Sie mehr über Astra Trident

Falls Sie neu bei Astra Trident sind, können Sie sich über die folgenden Links mit den folgenden Links vertraut machen:

- "FAQs"
- "Anforderungen für die Verwendung von Astra Trident"
- "Implementieren Sie Astra Trident"
- "Best Practices für die Konfiguration von ONTAP, Cloud Volumes ONTAP und Amazon FSX für NetApp ONTAP"
- "Integration Von Astra Trident"
- "Back-End-Konfiguration für ONTAP SAN"
- "Back-End-Konfiguration für ONTAP NAS"

Erfahren Sie mehr über die Treiberfunktionen "Hier".

Amazon FSX für NetApp ONTAP "FabricPool" Für das Management von Storage Tiers benötigen. Diese ermöglicht die Speicherung von Daten in einer Tier, basierend darauf, ob häufig auf die Daten zugegriffen wird.

Astra Trident erwartet, dass er mithilfe des Clusters entweder als ONTAP- oder SVM-Administrator ausgeführt wird `fsxadmin` Benutzer oder A `vsadmin` SVM-Benutzer oder ein Benutzer mit einem anderen Namen und derselben Rolle. Der `fsxadmin` Der Benutzer ist ein eingeschränkter Ersatz für den `admin` Cluster-Benutzer. Astra Trident verwendet in der Regel das `admin` Cluster-Benutzer für ONTAP Implementierungen, die nicht von Amazon FSX stammen

Treiber

Sie können Astra Trident mithilfe der folgenden Treiber in Amazon FSX für NetApp ONTAP integrieren:

- `ontap-san`: Jedes bereitgestellte PV ist eine LUN innerhalb seines eigenen Amazon FSX für NetApp ONTAP Volume.
- `ontap-san-economy`: Jedes bereitgestellte PV ist eine LUN mit einer konfigurierbaren Anzahl an LUNs pro Amazon FSX für das NetApp ONTAP Volume.

- `ontap-nas`: Jedes bereitgestellte PV ist ein vollständiger Amazon FSX für NetApp ONTAP Volume.
- `ontap-nas-economy`: Jedes bereitgestellte PV ist ein qtree mit einer konfigurierbaren Anzahl von qtrees pro Amazon FSX für NetApp ONTAP Volume.
- `ontap-nas-flexgroup`: Jedes bereitgestellte PV ist ein vollständiger Amazon FSX für NetApp ONTAP FlexGroup Volume.

Authentifizierung

Astra Trident bietet zwei Authentifizierungsmodi:

- Anmeldeinformationsbasiert: Sie können den verwenden `fsxadmin` Benutzer für Ihr Dateisystem oder die `vsadmin` Benutzer für Ihre SVM konfiguriert. Wir empfehlen die Verwendung `vsadmin` Benutzer zum Konfigurieren des Back-End. Astra Trident kommuniziert mit dem FSX-Dateisystem mit diesem Benutzernamen und Passwort.
- Zertifikatsbasiert: Astra Trident kommuniziert mit der SVM auf Ihrem FSX Dateisystem mit einem Zertifikat, das auf Ihrer SVM installiert ist.

Weitere Informationen zur Authentifizierung finden Sie unter folgenden Links:

- "ONTAP-NAS"
- "ONTAP SAN"

Astra Trident auf EKS mit Amazon FSX für NetApp ONTAP implementieren und konfigurieren

Was Sie benötigen

- Ein vorhandener Amazon EKS-Cluster oder selbst verwalteter Kubernetes-Cluster mit `kubectl` Installiert.
- Ein vorhandenes Amazon FSX für NetApp ONTAP Filesystem und Storage Virtual Machine (SVM), die über die Worker-Nodes Ihres Clusters erreichbar ist.
- Worker-Nodes, die vorbereitet sind "NFS und/oder iSCSI".

> Achten Sie darauf, dass Sie die für Amazon Linux und Ubuntu erforderlichen Schritte zur Knotenvorbereitung befolgen "Amazon Machine Images" (Amis) je nach EKS AMI-Typ.

Weitere Informationen zu Astra Trident-Anforderungen finden Sie unter "Hier".

Schritte

1. Astra Trident lässt sich mit einer der folgenden Methoden implementieren: ../Trident-Get-Started/kubernetes-Deploy.HTML[Deployment methods^].
2. Konfigurieren Sie Astra Trident wie folgt:
   a. Sammeln Sie den Management-LIF-DNS-Namen Ihrer SVM. Suchen Sie zum Beispiel über die AWS CLI nach `DNSName` Eintrag unter `Endpoints` → `Management` Nach Ausführung des folgenden Befehls:

      ```
      aws fsx describe-storage-virtual-machines --region <file system region>
      ```

3. Erstellen und Installieren von Zertifikaten zur Authentifizierung Wenn Sie ein verwenden `ontap-san` Back-

End, siehe "Hier". Wenn Sie ein verwenden `ontap-nas` Back-End, siehe "Hier".

> Sie können sich bei Ihrem Dateisystem anmelden (zum Beispiel Zertifikate installieren) mit SSH von überall, wo Sie Ihr Dateisystem erreichen können. Verwenden Sie die `fsxadmin` Benutzer, das Kennwort, das Sie beim Erstellen Ihres Dateisystems konfiguriert haben, und der Management-DNS-Name von `aws fsx describe-file-systems`.

4. Erstellen Sie eine Backend-Datei mithilfe Ihrer Zertifikate und des DNS-Namens Ihrer Management LIF, wie im folgenden Beispiel dargestellt:

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "customBackendName",
  "managementLIF": "svm-XXXXXXXXXXXXXXXXX.fs-XXXXXXXXXXXXXXXXX.fsx.us-
east-2.aws.internal",
  "svm": "svm01",
  "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
  "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
  "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz",
 }
```

Informationen zum Erstellen von Back-Ends finden Sie unter folgenden Links:

- "Konfigurieren Sie ein Backend mit ONTAP-NAS-Treibern"
- "Konfigurieren Sie ein Backend mit ONTAP-SAN-Treibern"

> Geben Sie nicht an `dataLIF` Für das `ontap-san` Und `ontap-san-economy` Treiber für den Einsatz von Multipath durch Astra Trident

> Bei der Verwendung von Amazon FSX für NetApp ONTAP mit Astra Trident, das `limitAggregateUsage` Der Parameter funktioniert nicht mit dem `vsadmin` Und `fsxadmin` Benutzerkonten. Der Konfigurationsvorgang schlägt fehl, wenn Sie diesen Parameter angeben.

Führen Sie nach der Bereitstellung die Schritte aus, um ein zu erstellen "Storage-Klasse, Volumes bereitstellen und das Volume in einem POD mounten".

Weitere Informationen

- "Dokumentation zu Amazon FSX für NetApp ONTAP"
- "Blogbeitrag zu Amazon FSX für NetApp ONTAP"

Back-Ends mit kubectl erstellen

Ein Backend definiert die Beziehung zwischen Astra Trident und einem Storage-System. Er erzählt Astra Trident, wie man mit diesem Storage-System kommuniziert und wie Astra Trident Volumes darauf bereitstellen sollte. Nach der Installation von Astra Trident ist der nächste Schritt die Erstellung eines Backend. Der

`TridentBackendConfig` Mit Custom Resource Definition (CRD) können Sie Trident Back-Ends direkt über die Kubernetes Schnittstelle erstellen und managen. Dies können Sie mit tun `kubectl` Oder das vergleichbare CLI Tool für Ihre Kubernetes Distribution.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`) Ist ein Front-End, Namensvetter CRD, mit dem Sie Astra Trident Back-Ends mit verwalten können `kubectl`. Kubernetes- und Storage-Administratoren können Back-Ends jetzt direkt über die Kubernetes-CLI erstellen und managen, ohne dass ein dediziertes Dienstprogramm für die Befehlszeilenschnittstelle erforderlich ist (`tridentctl`).

Bei der Erstellung eines `TridentBackendConfig` Objekt, geschieht Folgendes:

- Ein Back-End wird automatisch von Astra Trident auf Basis der von Ihnen zu erstellenden Konfiguration erstellt. Dies wird intern als A dargestellt `TridentBackend` (`tbe`, `tridentbackend`) CR.
- Der `TridentBackendConfig` Ist eindeutig an A gebunden `TridentBackend` Das wurde von Astra Trident entwickelt.

Beide `TridentBackendConfig` Pflegt eine 1:1-Zuordnung mit einem `TridentBackend`. Die erstere Schnittstelle, die dem Benutzer zum Design und zur Konfiguration von Back-Ends zur Verfügung gestellt wird. Letztere ist, wie Trident das tatsächliche Backend-Objekt darstellt.

`TridentBackend` CRS werden automatisch von Astra Trident erstellt. Sie sollten diese nicht ändern. Wenn Sie an Back-Ends Aktualisierungen vornehmen möchten, ändern Sie das `TridentBackendConfig` Objekt:

Im folgenden Beispiel finden Sie Informationen zum Format des `TridentBackendConfig` CR:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Sie können sich auch die Beispiele im ansehen "trident-Installationsprogramm" Verzeichnis für Beispielkonfigurationen für die gewünschte Speicherplattform/den gewünschten Service.

Der `spec` Nimmt Back-End-spezifische Konfigurationsparameter ein. In diesem Beispiel verwendet das Backend `ontap-san` Speichertreiber und verwendet die hier tabellarischen Konfigurationsparameter. Eine Liste der Konfigurationsoptionen für den gewünschten Speichertreiber finden Sie unter "Back-End-

Konfigurationsinformationen für Ihren Speichertreiber".

Der `spec` Abschnitt enthält auch `credentials` Und `deletionPolicy` Felder, die neu in den eingeführt werden `TridentBackendConfig` CR:

- `credentials`: Dieser Parameter ist ein Pflichtfeld und enthält die Anmeldeinformationen, die zur Authentifizierung mit dem Speichersystem/Service verwendet werden. Dies ist auf ein vom Benutzer erstelltes Kubernetes Secret festgelegt. Die Anmeldeinformationen können nicht im Klartext weitergegeben werden und führen zu einem Fehler.
- `deletionPolicy`: Dieses Feld definiert, was passieren soll, wenn der `TridentBackendConfig` Wird gelöscht. Es kann einen von zwei möglichen Werten annehmen:
  - `delete`: Dies führt zur Löschung beider `TridentBackendConfig` CR und das zugehörige Backend. Dies ist der Standardwert.
  - `retain`: Wenn a `TridentBackendConfig` CR wird gelöscht, die Backend-Definition ist weiterhin vorhanden und kann mit verwaltet werden `tridentctl`. Einstellen der Löschrichtlinie auf `retain` Benutzer können ein Downgrade auf eine frühere Version (vor 21.04) durchführen und die erstellten Back-Ends behalten. Der Wert für dieses Feld kann nach einem aktualisiert werden `TridentBackendConfig` Wird erstellt.

Der Name eines Backend wird mit festgelegt `spec.backendName`. Wenn nicht angegeben, wird der Name des Backend auf den Namen des gesetzt `TridentBackendConfig` Objekt (metadata.name). Es wird empfohlen, mit explizit Back-End-Namen festzulegen `spec.backendName`.

Back-Ends, die mit erstellt wurden `tridentctl` Ist nicht zugeordnet `TridentBackendConfig` Objekt: Sie können solche Back-Ends mit verwalten `kubectl` Durch Erstellen von A `TridentBackendConfig` CR. Es muss sorgfältig darauf achten, identische Konfigurationsparameter festzulegen (z. B. `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`, Und so weiter). Astra Trident bindet automatisch die neu erstellte `TridentBackendConfig` Mit dem bereits vorhandenen Backend.

Schritte im Überblick

Um ein neues Backend mit zu erstellen `kubectl`, Sie sollten Folgendes tun:

1. Erstellen Sie ein "Kubernetes Secret". Das Geheimnis enthält die Zugangsdaten, die Astra Trident zur Kommunikation mit dem Storage-Cluster/Service benötigt.
2. Erstellen Sie ein `TridentBackendConfig` Objekt: Dies enthält Angaben zum Storage-Cluster/Service und verweist auf das im vorherigen Schritt erstellte Geheimnis.

Nachdem Sie ein Backend erstellt haben, können Sie den Status mit beobachten `kubectl get tbc <tbc-name> -n <trident-namespace>` Und sammeln Sie weitere Details.

Schritt: Ein Kubernetes Secret erstellen

Erstellen Sie einen geheimen Schlüssel, der die Anmeldedaten für den Zugriff für das Backend enthält. Dies ist nur bei jedem Storage Service/jeder Plattform möglich. Hier ein Beispiel:

```
$ kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: t@Ax@7q(>
```

In dieser Tabelle sind die Felder zusammengefasst, die für jede Speicherplattform im Secret enthalten sein müssen:

Beschreibung der geheimen Felder der Speicherplattform	Geheim	Feldbeschreibung
Azure NetApp Dateien	Client-ID	Die Client-ID aus einer App-Registrierung
Cloud Volumes Service für AWS	ApiKey	CVS-Konto-API-Schlüssel
Cloud Volumes Service für AWS	SecretKey	Geheimer Schlüssel für CVS-Konto
Cloud Volumes Service für GCP	Private_Schlüssel_id	ID des privaten Schlüssels. Teil des API-Schlüssels für GCP-Servicekonto mit CVS-Administratorrolle
Cloud Volumes Service für GCP	Privater_Schlüssel	Privater Schlüssel. Teil des API-Schlüssels für GCP-Servicekonto mit CVS-Administratorrolle
Element (NetApp HCI/SolidFire)	Endpunkt	MVIP für den SolidFire-Cluster mit Mandanten-Anmeldedaten
ONTAP	Benutzername	Benutzername für die Verbindung mit dem Cluster/SVM. Wird für die Anmeldeinformationsbasierte Authentifizierung verwendet
ONTAP	Passwort	Passwort für die Verbindung mit dem Cluster/SVM Wird für die Anmeldeinformationsbasierte Authentifizierung verwendet

Beschreibung der geheimen Felder der Speicherplattform	Geheim	Feldbeschreibung
ONTAP	KundenPrivateKey	Base64-kodierte Wert des privaten Client-Schlüssels. Wird für die zertifikatbasierte Authentifizierung verwendet
ONTAP	ChapUsername	Eingehender Benutzername. Erforderlich, wenn usCHAP=true verwendet wird. Für `ontap-san` Und `ontap-san-economy`
ONTAP	ChapInitiatorSecret	CHAP-Initiatorschlüssel. Erforderlich, wenn usCHAP=true verwendet wird. Für `ontap-san` Und `ontap-san-economy`
ONTAP	ChapTargetBenutzername	Zielbenutzername. Erforderlich, wenn usCHAP=true verwendet wird. Für `ontap-san` Und `ontap-san-economy`
ONTAP	ChapTargetInitiatorSecret	Schlüssel für CHAP-Zielinitiator. Erforderlich, wenn usCHAP=true verwendet wird. Für `ontap-san` Und `ontap-san-economy`

Auf das in diesem Schritt erstellte Geheimnis wird im verwiesen `spec.credentials` Feld von `TridentBackendConfig` Objekt, das im nächsten Schritt erstellt wird.

Schritt 2: Erstellen Sie die `TridentBackendConfig` CR

Sie sind jetzt bereit, Ihre zu erstellen `TridentBackendConfig` CR. In diesem Beispiel wird ein Backend verwendet, das den verwendet `ontap-san` Treiber wird mithilfe des erstellt `TridentBackendConfig` Unten gezeigte Objekte:

```
$ kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Schritt 3: Überprüfen Sie den Status des `TridentBackendConfig` CR

Nun, da Sie die erstellt haben `TridentBackendConfig` CR, Sie können den Status überprüfen. Das folgende Beispiel zeigt:

```
$ kubectl -n trident get tbc backend-tbc-ontap-san
NAME                     BACKEND NAME          BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san   ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success
```

Ein Back-End wurde erfolgreich erstellt und an das gebunden `TridentBackendConfig` CR.

Die Phase kann einen der folgenden Werte annehmen:

- `Bound`: Das `TridentBackendConfig` CR ist mit einem Backend verknüpft, und dieses Backend enthält `configRef` Auf einstellen `TridentBackendConfig` CR's uid.
- `Unbound`: Dargestellt mit `""`. Der `TridentBackendConfig` Objekt ist nicht an ein Backend gebunden. Neu erstellt `TridentBackendConfig` CRS befinden sich standardmäßig in dieser Phase. Wenn die Phase sich ändert, kann sie nicht wieder auf Unbound zurückgesetzt werden.
- `Deleting`: Das `TridentBackendConfig` CR's `deletionPolicy` Wurde auf Löschen festgelegt. Wenn der `TridentBackendConfig` CR wird gelöscht und wechselt in den Löschzustand.
  - Wenn im Backend keine PVCs (Persistent Volume Claims) vorhanden sind, löschen Sie den `TridentBackendConfig` Wird dazu führen, dass Astra Trident das Backend sowie das löscht `TridentBackendConfig` CR.
  - Wenn ein oder mehrere VES im Backend vorhanden sind, wechselt es in den Löschzustand. Der `TridentBackendConfig` Anschließend wechselt CR in die Löschphase. Das Backend und `TridentBackendConfig` Werden erst gelöscht, nachdem alle PVCs gelöscht wurden.
- `Lost`: Das Backend, das mit dem verbunden ist `TridentBackendConfig` CR wurde versehentlich oder absichtlich gelöscht und das `TridentBackendConfig` CR hat noch einen Verweis auf das gelöschte

Backend. Der `TridentBackendConfig` CR kann weiterhin unabhängig vom gelöscht werden `deletionPolicy` Wert:

- `Unknown`: Astra Trident kann den Zustand oder die Existenz des mit dem verbundenen Backend nicht bestimmen `TridentBackendConfig` CR. Beispiel: Wenn der API-Server nicht antwortet oder wenn der `tridentbackends.trident.netapp.io` CRD fehlt. Dies kann eine Intervention des Benutzers erfordern.

In dieser Phase wird erfolgreich ein Backend erstellt! Es gibt mehrere Operationen, die zusätzlich gehandhabt werden können, wie z. B. "Back-End-Updates und Löschungen am Back-End".

(Optional) Schritt 4: Weitere Informationen

Sie können den folgenden Befehl ausführen, um weitere Informationen über Ihr Backend zu erhalten:

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

```
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success   ontap-san        delete
```

Zusätzlich können Sie auch einen YAML/JSON Dump von erhalten `TridentBackendConfig`.

```
$ kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: "2021-04-21T20:45:11Z"
  finalizers:
  - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound
```

`backendInfo` Enthält `backendName` Und das `backendUUID` Des Back-End, das als Antwort auf das erstellt wurde `TridentBackendConfig` CR. Der `lastOperationStatus` Feld gibt den Status des letzten Vorgangs des an `TridentBackendConfig` CR, der vom Benutzer ausgelöst werden kann (z. B. hat der Benutzer etwas in geändert `spec`) Oder ausgelöst durch Astra Trident (z. B. während Astra Trident Neustart). Er kann entweder erfolgreich oder fehlgeschlagen sein. `phase` Stellt den Status der Beziehung zwischen dem dar `TridentBackendConfig` CR und das Backend. Im obigen Beispiel `phase` Hat den Wert gebunden, was bedeutet, dass der `TridentBackendConfig` CR ist mit dem Backend verknüpft.

Sie können die ausführen `kubectl -n trident describe tbc <tbc-cr-name>` Befehl, um Details zu den Ereignisprotokollen zu erhalten.

Sie können ein Back-End, das einen zugeordneten enthält, nicht aktualisieren oder löschen `TridentBackendConfig` Objekt wird verwendet `tridentctl`. Um die Schritte zu verstehen, die mit dem Wechsel zwischen verbunden sind `tridentctl` Und `TridentBackendConfig`, "Sehen Sie hier".

Führen Sie das Back-End-Management mit kubectl durch

Erfahren Sie, wie Sie mit Backend-Management-Operationen durchführen `kubectl`.

Löschen Sie ein Back-End

Durch Löschen von A `TridentBackendConfig`, Sie weisen Astra Trident an, Back-Ends zu löschen/zu behalten (basierend auf `deletionPolicy`). Um ein Backend zu löschen, stellen Sie sicher, dass `deletionPolicy` Ist auf Löschen festgelegt. Um nur die zu löschen `TridentBackendConfig`, Stellen Sie das sicher `deletionPolicy` Auf beibehalten eingestellt. Dadurch wird sichergestellt, dass das Backend weiterhin vorhanden ist und mit verwaltet werden kann `tridentctl`.

Führen Sie den folgenden Befehl aus:

```
$ kubectl delete tbc <tbc-name> -n trident
```

Astra Trident löscht nicht die Kubernetes Secrets, die von verwendet wurden `TridentBackendConfig`. Der Kubernetes-Benutzer ist für die Bereinigung von Geheimnissen verantwortlich. Beim Löschen von Geheimnissen ist Vorsicht zu nehmen. Sie sollten Geheimnisse nur löschen, wenn sie nicht von den Back-Ends verwendet werden.

Zeigen Sie die vorhandenen Back-Ends an

Führen Sie den folgenden Befehl aus:

```
$ kubectl get tbc -n trident
```

Sie können auch ausführen `tridentctl get backend -n trident` Oder `tridentctl get backend -o yaml -n trident` Um eine Liste aller vorhandenen Back-Ends zu erhalten. Diese Liste umfasst auch Back-Ends, die mit erstellt wurden `tridentctl`.

Aktualisieren Sie ein Backend

Es gibt mehrere Gründe für die Aktualisierung eines Backend:

- Die Anmeldeinformationen für das Speichersystem wurden geändert. Um Anmeldedaten zu aktualisieren, wird das in verwendete Kubernetes Secret verwendet `TridentBackendConfig` Objekt muss aktualisiert werden. Astra Trident aktualisiert automatisch das Backend mit den neuesten Zugangsdaten. Führen Sie den folgenden Befehl aus, um den Kubernetes Secret zu aktualisieren:

```
$ kubectl apply -f <updated-secret-file.yaml> -n trident
```

- Parameter (wie der Name der verwendeten ONTAP-SVM) müssen aktualisiert werden. In diesem Fall `TridentBackendConfig` Objekte können direkt über Kubernetes aktualisiert werden.

```
$ kubectl apply -f <updated-backend-file.yaml>
```

Alternativ können Sie Änderungen an der vorhandenen vornehmen `TridentBackendConfig` CR durch Ausführen des folgenden Befehls:

```
$ kubectl edit tbc <tbc-name> -n trident
```

Wenn ein Backend-Update fehlschlägt, bleibt das Backend in seiner letzten bekannten Konfiguration erhalten. Sie können die Protokolle anzeigen, um die Ursache durch Ausführen zu bestimmen `kubectl get tbc <tbc-name> -o yaml -n trident` Oder `kubectl describe tbc <tbc-name> -n trident`.

Nachdem Sie das Problem mit der Konfigurationsdatei erkannt und behoben haben, können Sie den Befehl Update erneut ausführen.

Back-End-Management mit tridentctl

Erfahren Sie, wie Sie mit Backend-Management-Operationen durchführen `tridentctl`.

Erstellen Sie ein Backend

Nachdem Sie ein erstellt haben "Back-End-Konfigurationsdatei", Ausführen des folgenden Befehls:

```
$ tridentctl create backend -f <backend-file> -n trident
```

Wenn die Back-End-Erstellung fehlschlägt, ist mit der Back-End-Konfiguration ein Fehler aufgetreten. Sie können die Protokolle zur Bestimmung der Ursache anzeigen, indem Sie den folgenden Befehl ausführen:

```
$ tridentctl logs -n trident
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und behoben haben, können Sie einfach die ausführen `create` Befehl erneut.

Löschen Sie ein Back-End

Gehen Sie wie folgt vor, um ein Backend von Astra Trident zu löschen:

1. Abrufen des Back-End-Namens:

   ```
   $ tridentctl get backend -n trident
   ```

2. Back-End löschen:

```
$ tridentctl delete backend <backend-name> -n trident
```

Wenn Astra Trident Volumes und Snapshots aus diesem Backend bereitgestellt hat, die immer noch vorhanden sind, verhindert das Löschen des Backend, dass neue Volumes bereitgestellt werden. Das Backend wird weiterhin in einem „Deleting“ Zustand vorhanden sein und Trident wird weiterhin diese Volumes und Snapshots verwalten, bis sie gelöscht werden.

Zeigen Sie die vorhandenen Back-Ends an

Gehen Sie zum Anzeigen der von Trident verwendeten Back-Ends wie folgt vor:

- Führen Sie den folgenden Befehl aus, um eine Zusammenfassung anzuzeigen:

```
$ tridentctl get backend -n trident
```

- Um alle Details anzuzeigen, führen Sie den folgenden Befehl aus:

```
$ tridentctl get backend -o json -n trident
```

Aktualisieren Sie ein Backend

Führen Sie nach dem Erstellen einer neuen Backend-Konfigurationsdatei den folgenden Befehl aus:

```
$ tridentctl update backend <backend-name> -f <backend-file> -n trident
```

Wenn das Backend-Update fehlschlägt, ist bei der Backend-Konfiguration ein Fehler aufgetreten oder Sie haben ein ungültiges Update versucht. Sie können die Protokolle zur Bestimmung der Ursache anzeigen, indem Sie den folgenden Befehl ausführen:

```
$ tridentctl logs -n trident
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und behoben haben, können Sie einfach die ausführen `update` Befehl erneut.

Identifizieren Sie die Storage-Klassen, die ein Backend nutzen

Dies ist ein Beispiel für die Art von Fragen, die Sie mit der JSON beantworten können `tridentctl` Ausgänge für Backend-Objekte. Dazu wird der verwendet `jq` Dienstprogramm, das Sie installieren müssen.

```
$ tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

Dies gilt auch für Back-Ends, die mit erstellt wurden `TridentBackendConfig`.

Wechseln Sie zwischen den Back-End-Managementoptionen

Erfahren Sie in Astra Trident, wie Back-Ends auf verschiedene Art und Weise gemanagt werden. Mit der Einführung von `TridentBackendConfig`, Administratoren haben jetzt zwei unterschiedliche Arten von Back-Ends zu verwalten. Dies stellt die folgenden Fragen:

- Mit können Back-Ends erstellt werden `tridentctl` Gemanagt werden mit `TridentBackendConfig`?
- Mit können Back-Ends erstellt werden `TridentBackendConfig` Gemanagt werden mit `tridentctl`?

Managen `tridentctl` Back-Ends mit `TridentBackendConfig`

In diesem Abschnitt werden die Schritte aufgeführt, die für das Management von Back-Ends erforderlich sind, die mit erstellt wurden `tridentctl` Erstellen Sie direkt über die Kubernetes Schnittstelle `TridentBackendConfig` Objekte:

Dies gilt für die folgenden Szenarien:

- Bereits vorhandene Back-Ends, die kein A haben `TridentBackendConfig` Weil sie mit erstellt wurden `tridentctl`.
- Neue Back-Ends, mit denen erstellt wurden `tridentctl`, Während andere `TridentBackendConfig` Objekte sind vorhanden.

In beiden Szenarien werden Back-Ends weiterhin vorhanden sein, wobei Astra Trident Volumes terminieren und darauf arbeiten wird. Administratoren können hier eine von zwei Möglichkeiten wählen:

- Fahren Sie mit der Verwendung fort `tridentctl` Um Back-Ends zu managen, die mit ihr erstellt wurden.
- Back-Ends werden mit erstellt `tridentctl` Zu einer neuen `TridentBackendConfig` Objekt: Dies würde bedeuten, dass die Back-Ends mit gemanagt werden `kubectl` Und nicht `tridentctl`.

Um ein bereits vorhandenes Backend mit zu verwalten `kubectl`, Sie müssen ein erstellen `TridentBackendConfig` Das bindet an das vorhandene Backend. Hier eine Übersicht über die Funktionsweise:

1. Kubernetes Secret erstellen: Das Geheimnis enthält die Zugangsdaten, die Astra Trident zur Kommunikation mit dem Storage-Cluster/Service benötigt.
2. Erstellen Sie ein `TridentBackendConfig` Objekt: Dies enthält Angaben zum Storage-Cluster/Service und verweist auf das im vorherigen Schritt erstellte Geheimnis. Es muss sorgfältig darauf achten, identische Konfigurationsparameter festzulegen (z. B. `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`, Und so weiter). `spec.backendName` Muss auf den Namen des vorhandenen Backend eingestellt werden.

Schritt 0: Identifizieren Sie das Backend

Um ein zu erstellen `TridentBackendConfig` Das bindet an ein vorhandenes Backend, müssen Sie die Konfiguration des Backend erhalten. In diesem Beispiel nehmen wir an, dass ein Backend mithilfe der folgenden JSON-Definition erstellt wurde:

```
$ tridentctl get backend ontap-nas-backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+

$ cat ontap-nas-backend.json

{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.10.10.1",
    "dataLIF": "10.10.10.2",
    "backendName": "ontap-nas-backend",
    "svm": "trident_svm",
    "username": "cluster-admin",
    "password": "admin-password",

    "defaults": {
        "spaceReserve": "none",
        "encryption": "false"
    },
    "labels":{"store":"nas_store"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"app":"msoffice", "cost":"100"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"app":"mysqldb", "cost":"25"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
```

```
                "unixPermissions": "0775"
            }
        }
    ]
}
```

Schritt: Ein Kubernetes Secret erstellen

Erstellen Sie einen geheimen Schlüssel, der die Anmeldeinformationen für das Backend enthält, wie in diesem Beispiel gezeigt:

```
$ cat tbc-ontap-nas-backend-secret.yaml

apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  passWord: admin-password

$ kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

Schritt 2: Erstellen Sie ein `TridentBackendConfig` CR

Im nächsten Schritt wird ein erstellt `TridentBackendConfig` CR, das automatisch an die bereits vorhandene bindet `ontap-nas-backend` (Wie in diesem Beispiel). Stellen Sie sicher, dass folgende Anforderungen erfüllt sind:

- Der gleiche Backend-Name wird in definiert `spec.backendName`.
- Die Konfigurationsparameter sind mit dem ursprünglichen Back-End identisch.
- Virtual Storage Pools (falls vorhanden) müssen dieselbe Reihenfolge beibehalten wie im ursprünglichen Backend.
- Anmeldedaten werden bei einem Kubernetes Secret und nicht im Klartext bereitgestellt.

In diesem Fall die `TridentBackendConfig` Wird so aussehen:

```
$ cat backend-tbc-ontap-nas.yaml
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
      app: msoffice
      cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
      app: mysqldb
      cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'

$ kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

Schritt 3: Überprüfen Sie den Status des `TridentBackendConfig` CR

Nach dem `TridentBackendConfig` Wurde erstellt, seine Phase muss sein `Bound`. Sie sollte außerdem den gleichen Backend-Namen und die gleiche UUID wie das vorhandene Backend widerspiegeln.

```
$ kubectl -n trident get tbc tbc-ontap-nas-backend -n trident
NAME                   BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
$ tridentctl get backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

Das Backend wird nun vollständig mit dem verwaltet `tbc-ontap-nas-backend` `TridentBackendConfig` Objekt:

Managen `TridentBackendConfig` Back-Ends mit `tridentctl`

```
`tridentctl` Kann zur Auflistung von Back-Ends verwendet werden, die mit
erstellt wurden `TridentBackendConfig`. Darüber hinaus können
Administratoren solche Back-Ends mithilfe von auch vollständig managen
`tridentctl` Durch Löschen `TridentBackendConfig` Mit Sicherheit
`spec.deletionPolicy` Ist auf festgelegt `retain`.
```

Schritt 0: Identifizieren Sie das Backend

Nehmen wir beispielsweise an, dass das folgende Backend mit erstellt wurde `TridentBackendConfig`:

```
$ kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

$ tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

Von der Ausgabe, ist es gesehen, dass `TridentBackendConfig` Wurde erfolgreich erstellt und ist an ein Backend gebunden [die UUID des Backend beachten].

Schritt 1: Bestätigen `deletionPolicy` **Ist auf festgelegt** `retain`

Lassen Sie uns den Wert von betrachten `deletionPolicy`. Dies muss eingestellt werden `retain`. Dadurch wird sichergestellt, dass, wenn ein `TridentBackendConfig` CR wird gelöscht, die Backend-Definition ist weiterhin vorhanden und kann mit verwaltet werden `tridentctl`.

```
$ kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
$ kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
$ kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

Fahren Sie nur mit dem nächsten Schritt fort `deletionPolicy` Ist auf festgelegt `retain`.

Schritt 2: Löschen Sie den `TridentBackendConfig` CR

Der letzte Schritt besteht darin, den zu löschen `TridentBackendConfig` CR. Nach Bestätigung des `deletionPolicy` Ist auf festgelegt `retain`, Sie können mit der Löschung fortfahren:

```
$ kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

$ tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE   | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

Bei der Löschung der `TridentBackendConfig` Object, Astra Trident entfernt es einfach, ohne das Backend zu löschen.

Management von Storage-Klassen

Suchen Sie Informationen zum Erstellen einer Speicherklasse, Löschen einer Speicherklasse und Anzeigen vorhandener Speicherklassen.

Entwurf einer Storage-Klasse

Siehe "Hier" Finden Sie weitere Informationen darüber, welche Storage-Klassen es gibt und wie Sie sie konfigurieren.

Erstellen Sie eine Speicherklasse

Führen Sie den folgenden Befehl aus, wenn Sie eine Speicherklassendatei haben:

```
kubectl create -f <storage-class-file>
```

`<storage-class-file>` Sollte durch den Dateinamen der Speicherklasse ersetzt werden.

Löschen Sie eine Speicherklasse

Führen Sie den folgenden Befehl aus, um eine Storage-Klasse aus Kubernetes zu löschen:

```
kubectl delete storageclass <storage-class>
```

`<storage-class>` Sollten durch Ihre Storage-Klasse ersetzt werden.

Alle persistenten Volumes, die durch diese Storage-Klasse erstellt wurden, werden unverändert beibehalten und Astra Trident wird sie weiterhin managen.

> Astra Trident setzt ein Leereinschub um `fsType` Für die von ihm erstellten Volumes. Bei iSCSI-Back-Ends wird die Durchsetzung empfohlen `parameters.fsType` In der StorageClass. Sie sollten esixting StorageClasses löschen und mit neu erstellen `parameters.fsType` Angegeben.

Sehen Sie sich die vorhandenen Speicherklassen an

- Um vorhandene Kubernetes-Storage-Klassen anzuzeigen, führen Sie den folgenden Befehl aus:

```
kubectl get storageclass
```

- Um die Details der Kubernetes-Storage-Klasse anzuzeigen, führen Sie den folgenden Befehl aus:

```
kubectl get storageclass <storage-class> -o json
```

- Führen Sie den folgenden Befehl aus, um die synchronisierten Storage-Klassen von Astra Trident anzuzeigen:

```
tridentctl get storageclass
```

- Um die synchronisierten Storage-Klassendetails von Astra Trident anzuzeigen, führen Sie den folgenden Befehl aus:

```
tridentctl get storageclass <storage-class> -o json
```

Legen Sie eine Standardspeicherklasse fest

Mit Kubernetes 1.6 können Sie eine Standard-Storage-Klasse festlegen. Dies ist die Storage-Klasse, die zur Bereitstellung eines Persistent Volume verwendet wird, wenn ein Benutzer in einer Persistent Volume Claim (PVC) nicht eine Angabe vorgibt.

- Definieren Sie eine Standard-Storage-Klasse, indem Sie die Anmerkung festlegen `storageclass.kubernetes.io/is-default-class` In der Definition der Storage-Klassen wie den

„true“. Gemäß der Spezifikation wird jeder andere Wert oder jede Abwesenheit der Anmerkung als falsch interpretiert.

- Sie können eine vorhandene Storage-Klasse als Standard-Storage-Klasse konfigurieren, indem Sie den folgenden Befehl verwenden:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- In ähnlicher Weise können Sie die standardmäßige Storage-Klassenbeschriftung mithilfe des folgenden Befehls entfernen:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Es gibt auch Beispiele im Trident Installationspaket, die diese Annotation enthält.

Sie sollten immer nur eine Standard-Storage-Klasse in Ihrem Cluster verwenden. Kubernetes verhindert technisch nicht, dass Sie mehr als eine haben, aber es verhält sich so, als ob es überhaupt keine Standard-Storage-Klasse gibt.

Das Backend für eine Storage-Klasse ermitteln

Dies ist ein Beispiel für die Art von Fragen, die Sie mit der JSON beantworten können `tridentctl` Ausgänge für Astra Trident Backend-Objekte. Dazu wird der verwendet `jq` Dienstprogramm, das Sie möglicherweise zuerst installieren müssen.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

Durchführung von Volume-Vorgängen

Erfahren Sie mehr über die Funktionen von Astra Trident zum Management Ihrer Volumes.

- "Verwenden Sie die CSI-Topologie"
- "Arbeiten Sie mit Snapshots"
- "Erweitern Sie Volumes"
- "Volumes importieren"

Verwenden Sie die CSI-Topologie

Astra Trident kann Volumes selektiv erstellen und zu Nodes in einem Kubernetes Cluster verbinden, indem der verwendet wird "Funktion CSI Topology". Mithilfe der CSI Topology-Funktion kann der Zugriff auf Volumes auf einen Teil von Nodes basierend auf Regionen und Verfügbarkeitszonen begrenzt werden. Cloud-Provider ermöglichen Kubernetes-Administratoren inzwischen das Erstellen von Nodes, die zonenbasiert sind. Die Nodes können sich in verschiedenen Regionen innerhalb einer Verfügbarkeitszone oder über verschiedene

Verfügbarkeitszonen hinweg befinden. Astra Trident verwendet CSI Topology, um die Provisionierung von Volumes für Workloads in einer Multi-Zone-Architektur zu vereinfachen.

Erfahren Sie mehr über die Funktion CSI Topology "Hier".

Kubernetes bietet zwei unterschiedliche Modi für die Volume-Bindung:

- Mit `VolumeBindingMode` Auf einstellen `Immediate`, Astra Trident erstellt das Volume ohne Topologiebewusstsein. Die Volume-Bindung und die dynamische Bereitstellung werden bei der Erstellung des PVC behandelt. Dies ist die Standardeinstellung `VolumeBindingMode` Und ist für Cluster geeignet, die keine Topologiebeschränkungen mehr durchsetzen. Persistente Volumes werden erstellt, ohne dass sie von den Planungsanforderungen des anfragenden Pods abhängig sind.
- Mit `VolumeBindingMode` Auf einstellen `WaitForFirstConsumer`, Die Erstellung und Bindung eines Persistent Volume für ein PVC wird verzögert, bis ein Pod, der die PVC verwendet, geplant und erstellt wird. Auf diese Weise werden Volumes erstellt, um Planungseinschränkungen zu erfüllen, die durch Topologieanforderungen durchgesetzt werden.

Der `WaitForFirstConsumer` Für den Bindungsmodus sind keine Topologiebeschriftungen erforderlich. Diese kann unabhängig von der CSI Topology Funktion verwendet werden.

Was Sie benötigen

Für die Verwendung von CSI Topology benötigen Sie Folgendes:

- Kubernetes-Cluster mit 1.17 oder höher

```
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- Nodes im Cluster sollten über Labels verfügen, die eine Topologiebewusstsein einführen (`topology.kubernetes.io/region` Und `topology.kubernetes.io/zone`). Diese Labels * sollten auf Knoten im Cluster vorhanden sein* bevor Astra Trident installiert ist, damit Astra Trident Topologieorientiert ist.

```
$ kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

Schritt 1: Erstellen Sie ein Topologieorientiertes Backend

Astra Trident Storage-Back-Ends können für die selektive Bereitstellung von Volumes basierend auf Verfügbarkeitszonen ausgelegt werden. Jedes Backend kann optional mittragen `supportedTopologies` Block, der eine Liste der zu unterstützenden Zonen und Regionen darstellt. Bei StorageClasses, die ein solches Backend nutzen, wird ein Volume nur erstellt, wenn es von einer Applikation angefordert wird, die in einer unterstützten Region/Zone geplant ist.

So sieht eine Beispiel-Backend-Definition aus:

```
{
 "version": 1,
 "storageDriverName": "ontap-san",
 "backendName": "san-backend-us-east1",
 "managementLIF": "192.168.27.5",
 "svm": "iscsi_svm",
 "username": "admin",
 "password": "xxxxxxxxxxxx",
 "supportedTopologies": [
{"topology.kubernetes.io/region": "us-east1",
"topology.kubernetes.io/zone": "us-east1-a"},
{"topology.kubernetes.io/region": "us-east1",
"topology.kubernetes.io/zone": "us-east1-b"}
]
}
```

`supportedTopologies` Wird verwendet, um eine Liste von Regionen und Zonen pro Backend bereitzustellen. Diese Regionen und Zonen stellen die Liste der zulässigen Werte dar, die in einer StorageClass bereitgestellt werden können. Bei StorageClasses, die einen Teil der Regionen und Zonen enthalten, die in einem Backend bereitgestellt werden, erstellt Astra Trident ein Volume im Backend.

Sie können definieren `supportedTopologies` Auch pro Storagepool. Das folgende Beispiel zeigt:

```
{"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "nas-backend-us-central1",
"managementLIF": "172.16.238.5",
"svm": "nfs_svm",
"username": "admin",
"password": "Netapp123",
"supportedTopologies": [
      {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-a"},
      {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-b"}
    ]
"storage": [
   {
       "labels": {"workload":"production"},
        "region": "Iowa-DC",
        "zone": "Iowa-DC-A",
        "supportedTopologies": [
            {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-a"}
        ]
    },
    {
        "labels": {"workload":"dev"},
         "region": "Iowa-DC",
         "zone": "Iowa-DC-B",
         "supportedTopologies": [
             {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-b"}
         ]
     }
]
}
```

In diesem Beispiel ist der `region` Und `zone` Etiketten stehen für die Position des Speicherpools. `topology.kubernetes.io/region` Und `topology.kubernetes.io/zone` Vorgeben, woher die Speicherpools verbraucht werden können.

Schritt: Definition von StorageClasses, die sich der Topologie bewusst sind

Auf der Grundlage der Topologiebeschriftungen, die den Nodes im Cluster zur Verfügung gestellt werden, können StorageClasses so definiert werden, dass sie Topologieinformationen enthalten. So werden die Storage-Pools festgelegt, die als Kandidaten für PVC-Anfragen dienen, und die Untergruppe der Nodes, die die von Trident bereitgestellten Volumes nutzen können.

Das folgende Beispiel zeigt:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
- key: topology.kubernetes.io/zone
  values:
  - us-east1-a
  - us-east1-b
- key: topology.kubernetes.io/region
  values:
  - us-east1
parameters:
  fsType: "ext4"
```

In der oben angegebenen StorageClass-Definition `volumeBindingMode` Ist auf festgelegt `WaitForFirstConsumer`. VES, die mit dieser StorageClass angefordert werden, werden erst dann gehandelt, wenn sie in einem Pod referenziert werden. Und `allowedTopologies` Stellt die Zonen und die Region bereit, die verwendet werden sollen. Der `netapp-san-us-east1` StorageClass erstellt VES auf dem `san-backend-us-east1` Back-End oben definiert.

Schritt 3: Erstellen und verwenden Sie ein PVC

Wenn die StorageClass erstellt und einem Backend zugeordnet wird, können Sie jetzt PVCs erstellen.

Siehe Beispiel `spec` Unten:

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

Das Erstellen eines PVC mithilfe dieses Manifests würde Folgendes zur Folge haben:

```
$ kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
$ kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
$ kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

Verwenden Sie für Trident, ein Volume zu erstellen und es an die PVC zu binden, das in einem Pod verwendet wird. Das folgende Beispiel zeigt:

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

Diese PodSpec beauftragt Kubernetes, den Pod auf Nodes zu planen, die in vorhanden sind `us-east1` Wählen Sie einen beliebigen Knoten aus, der im vorhanden ist `us-east1-a` Oder `us-east1-b` Zonen:

Siehe die folgende Ausgabe:

```
$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
$ kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

Aktualisieren Sie Back-Ends, um einzuschließen `supportedTopologies`

Vorhandene Back-Ends können mit einer Liste von aktualisiert werden `supportedTopologies` Wird verwendet `tridentctl backend update`. Dies wirkt sich nicht auf Volumes aus, die bereits bereitgestellt wurden und nur für nachfolgende VES verwendet werden.

Weitere Informationen

- "Management von Ressourcen für Container"
- "NodeSelector"
- "Affinität und Antiaffinität"
- "Tönungen und Tolerationen"

Arbeiten Sie mit Snapshots

Ab Version 20.01 von Astra Trident können Sie auf der Kubernetes-Ebene Snapshots von PVS erstellen. Mithilfe dieser Snapshots können zeitpunktgenaue Kopien von Volumes erstellt werden, die durch Astra Trident erstellt wurden, und die Erstellung zusätzlicher Volumes (Klone) geplant werden. Volume Snapshot wird von unterstützt `ontap-nas`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `aws-cvs`, `gcp-cvs`, und `azure-netapp-files` Treiber.

> Diese Funktion ist über Kubernetes 1.17 (Beta) erhältlich und wird ab 1.20 verfügbar sein. Informationen zu den Änderungen, die beim Wechsel von der Beta zu GA erforderlich sind, finden Sie unter "Der Release Blog". Mit der Graduierung zu GA, die `v1` Die API-Version wird eingeführt und ist abwärtskompatibel mit `v1beta1` Snapshots:

Was Sie benötigen

- Zum Erstellen von Volume-Snapshots müssen sowohl ein externer Snapshot-Controller als auch einige Custom Resource Definitions (CRDs) erstellt werden. In dieser Verantwortung liegt der aktuell verwendete Kubernetes Orchestrator (z. B. Kubeadm, GKE, OpenShift).

Sie können einen externen Snapshot-Controller und Snapshot-CRDs wie folgt erstellen:

1. Erstellen von Volume-Snapshot-CRDs:

```
$ cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Erstellen Sie den Snapshot-Controller im gewünschten Namespace. Bearbeiten Sie die YAML-Manifeste unten, um den Namespace zu ändern.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-3.0/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-3.0/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

CSI-Snapshotter bietet ein "Webhook wird überprüft" Um Benutzern zu helfen, vorhandene v1beta1 Snapshots zu validieren und zu bestätigen, dass es sich um gültige Ressourcenobjekte handelt. Der validierende Webhook markiert automatisch ungültige Snapshot-Objekte und verhindert die Erstellung von zukünftigen ungültigen Objekten. Der validierende Webhook wird über den Kubernetes Orchestrator implementiert. Lesen Sie die Anweisungen, um den zu validierende Webhook manuell bereitzustellen "Hier". Beispiele für ungültige Snapshot-Manifeste "Hier".

Das unten aufgeführte Beispiel erläutert die Konstrukte, die für die Arbeit mit Snapshots erforderlich sind und zeigt, wie Snapshots erstellt und verwendet werden können.

Schritt 1: Richten Sie ein `VolumeSnapshotClass`

Richten Sie vor dem Erstellen eines Volume-Snapshots einen Link:./Trident-Referenz/objects.html ein[`VolumeSnapshotClass`^].

```
$ cat snap-sc.yaml
#Use apiVersion v1 for Kubernetes 1.20 and above. For Kubernetes 1.17 -
1.19, use apiVersion v1beta1.
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

Der `driver` Zeigt auf den CSI-Treiber von Astra Trident. `deletionPolicy` Kann sein `Delete` Oder `Retain`. Wenn eingestellt auf `Retain`, Der zugrunde liegende physische Snapshot auf dem Storage-Cluster wird auch dann beibehalten, wenn der `VolumeSnapshot` Objekt wurde gelöscht.

Schritt 2: Erstellen Sie einen Schnappschuss eines vorhandenen PVC

```
$ cat snap.yaml
#Use apiVersion v1 for Kubernetes 1.20 and above. For Kubernetes 1.17 -
1.19, use apiVersion v1beta1.
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

Der Snapshot wird für ein PVC mit dem Namen erstellt `pvc1`, Und der Name des Snapshots ist auf festgelegt `pvc1-snap`.

```
$ kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

$ kubectl get volumesnapshots
NAME                   AGE
pvc1-snap              50s
```

Dadurch wurde ein erstellt `VolumeSnapshot` Objekt: Ein VolumeSnapshot ist analog zu einem PVC und einem zugeordnet `VolumeSnapshotContent` Objekt, das den tatsächlichen Snapshot darstellt.

Es ist möglich, die zu identifizieren `VolumeSnapshotContent` Objekt für das `pvc1-snap` VolumeSnapshot wird beschrieben.

```
$ kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
.
.
.
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
.
.
```

Der `Snapshot Content Name` Identifiziert das VolumeSnapshotContent-Objekt, das diesen Snapshot bereitstellt. Der `Ready To Use` Der Parameter gibt an, dass der Snapshot zum Erstellen einer neuen PVC verwendet werden kann.

Schritt 3: PVCs aus VolumeSnapshots erstellen

Im folgenden Beispiel wird das Erstellen eines PVC mithilfe eines Snapshots beschrieben:

```
$ cat pvc-from-snap.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

`dataSource` Zeigt an, dass das PVC mit dem Namen VolumeSnapshot erstellt werden muss `pvc1-snap` Als Quelle der Daten. Damit beauftragt Astra Trident, aus dem Snapshot ein PVC zu erstellen. Nachdem die PVC erstellt wurde, kann sie an einem Pod befestigt und wie jedes andere PVC verwendet werden.

> Wenn Sie ein persistentes Volume mit zugeordneten Snapshots löschen, wird das entsprechende Trident-Volume in einen „Löschzustand“ aktualisiert. Damit das Astra Trident Volume gelöscht werden kann, sollten die Snapshots des Volume entfernt werden.

Weitere Informationen

- "Volume Snapshots"
- Link:../Trident-Referenz/objects.html[`VolumeSnapshotClass`^]

Erweitern Sie Volumes

Astra Trident bietet Kubernetes-Benutzern die Möglichkeit, ihre Volumes nach Erstellung zu erweitern. Hier finden Sie Informationen zu den erforderlichen Konfigurationen zum erweitern von iSCSI- und NFS-Volumes.

Erweitern Sie ein iSCSI-Volume

Sie können ein iSCSI Persistent Volume (PV) mithilfe der CSI-provisionierung erweitern.

> Die Erweiterung des iSCSI-Volumes wird von unterstützt `ontap-san`, `ontap-san-economy`, `solidfire-san` Treiber und erfordert Kubernetes 1.16 und höher.

Überblick

Die Erweiterung eines iSCSI-PV umfasst die folgenden Schritte:

- Bearbeiten der StorageClass-Definition zum Festlegen des `allowVolumeExpansion` Feld an `true`.
- Bearbeiten der PVC-Definition und Aktualisieren der `spec.resources.requests.storage` Um die neu gewünschte Größe zu reflektieren, die größer als die ursprüngliche Größe sein muss.
- Das Anbringen des PV muss an einen Pod angehängt werden, damit die Größe geändert werden kann. Beim Ändern der Größe eines iSCSI-PV gibt es zwei Szenarien:
  - Wenn das PV an einen POD angeschlossen ist, erweitert Astra Trident das Volume auf dem Storage-Back-End, setzt das Gerät neu ein und vergrößert das Dateisystem neu.
  - Bei dem Versuch, die Größe eines nicht angeschlossenen PV zu ändern, erweitert Astra Trident das Volume auf dem Storage-Backend. Nachdem die PVC an einen Pod gebunden ist, lässt Trident das Gerät neu in die Größe des Dateisystems einarbeiten. Kubernetes aktualisiert dann die PVC-Größe, nachdem der Expand-Vorgang erfolgreich abgeschlossen ist.

Das folgende Beispiel zeigt, wie die Erweiterung von iSCSI PVS funktioniert.

Schritt: Storage Class für Volume-Erweiterung konfigurieren

```
$ cat storageclass-ontapsan.yaml
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Bearbeiten Sie für eine bereits vorhandene StorageClass, um die einzuschließen `allowVolumeExpansion` Parameter.

Schritt 2: Erstellen Sie ein PVC mit der von Ihnen erstellten StorageClass

```
$ cat pvc-ontapsan.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Astra Trident erstellt ein persistentes Volume (PV) und verknüpft es mit dieser Persistent Volume Claim (PVC).

```
$ kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               10s
```

Schritt 3: Definieren Sie einen Behälter, der das PVC befestigt

In diesem Beispiel wird ein POD erstellt, der die verwendet `san-pvc`.

```
$  kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
centos-pod   1/1     Running   0          65s

$  kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    centos-pod
```

Schritt 4: Erweitern Sie das PV

Um die Größe des PV zu ändern, das von 1Gi auf 2Gi erstellt wurde, bearbeiten Sie die PVC-Definition und aktualisieren Sie die `spec.resources.requests.storage` Bis 2Gi.

```
$ kubectl edit pvc san-pvc
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 ...
```

Schritt 5: Validieren Sie die Erweiterung

Sie können die korrekte Ausführung der Erweiterung überprüfen, indem Sie die Größe der PVC, PV und des Astra Trident Volume überprüfen:

```
$ kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
$ tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Erweitern Sie ein NFS-Volume

Astra Trident unterstützt die Volume-Erweiterung für auf bereitgestellte NFS PVS `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `aws-cvs`, `gcp-cvs`, und `azure-netapp-files` Back-Ends:

Schritt: Storage Class für Volume-Erweiterung konfigurieren

Um die Größe eines NFS PV zu ändern, muss der Administrator zunächst die Storage-Klasse konfigurieren, um die Volume-Erweiterung durch Einstellen der zu ermöglichen `allowVolumeExpansion` Feld an `true`:

```
$ cat storageclass-ontapnas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

Wenn Sie bereits eine Storage-Klasse ohne diese Option erstellt haben, können Sie die vorhandene Storage-Klasse einfach mit bearbeiten `kubectl edit storageclass` Um eine Volume-Erweiterung zu ermöglichen.

Schritt 2: Erstellen Sie ein PVC mit der von Ihnen erstellten StorageClass

```
$ cat pvc-ontapnas.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Astra Trident sollte ein 20MiB NFS PV für diese PVC erstellen:

```
$ kubectl get pvc
NAME           STATUS   VOLUME
CAPACITY     ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

$ kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

Schritt 3: Erweitern Sie das PV

Um die Größe des neu erstellten 20MiB PV auf 1 gib zu ändern, bearbeiten Sie die PVC und den Satz `spec.resources.requests.storage` Bis 1 GB:

```
$ kubectl edit pvc ontapnas20mb
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
...
```

Schritt 4: Validieren Sie die Erweiterung

Sie können die korrekte Größenänderung validieren, indem Sie die Größe des PVC, des PV und des Astra Trident Volume überprüfen:

```
$ kubectl get pvc ontapnas20mb
NAME           STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi
RWO            ontapnas        4m44s

$ kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

$ tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n
trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Volumes importieren

Sie können vorhandene Storage Volumes mit als Kubernetes PV importieren `tridentctl import`.

Treiber, die den Volumenimport unterstützen

In dieser Tabelle sind die Treiber aufgeführt, die den Import von Volumes unterstützen, und die Version, in der sie eingeführt wurden.

Treiber	Freigabe
`ontap-nas`	19.04
`ontap-nas-flexgroup`	19.04
`solidfire-san`	19.04
`aws-cvs`	19.04

Treiber	Freigabe
azure-netapp-files	19.04
gcp-cvs	19.04
ontap-san	19.04

Warum sollte ich Volumes importieren?

Es gibt verschiedene Anwendungsfälle für den Import eines Volumes in Trident:

- Eine Anwendung erreichen und ihren vorhandenen Datensatz erneut verwenden
- Verwenden eines Klons eines Datensatzes für eine kurzlebige Anwendung
- Neuerstellung eines fehlerhaften Kubernetes-Clusters
- Migration von Applikationsdaten während der Disaster Recovery

Wie funktioniert der Import?

Die PVC-Datei (Persistent Volume Claim) wird vom Importprozess des Volumes zur Erstellung des PVC verwendet. Die PVC-Datei sollte mindestens die Felder Name, Namespace, accessModes und storageClassName enthalten, wie im folgenden Beispiel dargestellt.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```

Der `tridentctl` Der Client wird verwendet, um ein vorhandenes Storage Volume zu importieren. Trident importiert das Volume, indem Volume-Metadaten gespeichert und die PVC und das PV erstellt werden.

```
$ tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-
file>
```

Zum Importieren eines Storage-Volumes geben Sie den Namen des Astra Trident Backends mit dem Volume sowie den Namen an, der das Volume auf dem Storage eindeutig identifiziert (z. B. ONTAP FlexVol, Element Volume, CVS Volume Path). Das Storage-Volume muss Lese-/Schreibzugriff ermöglichen und über das angegebene Astra Trident-Back-End zugänglich sein. Der `-f` String Argument ist erforderlich und gibt den Pfad zur YAML- oder JSON-PVC-Datei an.

Erhält Astra Trident die Anfrage für das Importvolumen, wird die vorhandene Volume-Größe festgelegt und im

PVC festgelegt. Nachdem das Volumen vom Speichertreiber importiert wurde, wird das PV mit einem ClaimRef an die PVC erzeugt. Die Rückgewinnungsrichtlinie ist zunächst auf festgelegt `retain` Im PV. Nachdem Kubernetes die PVC und das PV erfolgreich bindet, wird die Zurückgewinnungsrichtlinie aktualisiert und an die Zurückgewinnungsrichtlinie der Storage-Klasse angepasst. Wenn die Richtlinie zur Zurückgewinnung der Storage-Klasse lautet `delete`, Das Speichervolumen wird gelöscht, wenn das PV gelöscht wird.

Wenn ein Volume mit dem importiert wird `--no-manage` Argument: Trident führt für den Lebenszyklus der Objekte keine zusätzlichen Operationen an der PVC oder PV durch. Da Trident PV- und PVC-Ereignisse für ignoriert `--no-manage` Objekte, das Speichervolumen wird nicht gelöscht, wenn das PV gelöscht wird. Andere Vorgänge, wie z. B. der Volume-Klon und die Volume-Größe, werden ebenfalls ignoriert. Diese Option ist nützlich, wenn Sie Kubernetes für Workloads in Containern verwenden möchten, aber ansonsten den Lebenszyklus des Storage Volumes außerhalb von Kubernetes managen möchten.

Der PVC und dem PV wird eine Anmerkung hinzugefügt, die einem doppelten Zweck dient, anzugeben, dass das Volumen importiert wurde und ob PVC und PV verwaltet werden. Diese Anmerkung darf nicht geändert oder entfernt werden.

Trident 19.07 und höher verarbeiten den Anhang von PVS und mountet das Volume im Rahmen des Imports. Bei Importen mit früheren Versionen von Astra Trident gibt es keine Vorgänge im Datenpfad. Der Volume-Import überprüft nicht, ob das Volume gemountet werden kann. Wenn beim Import des Volumes ein Fehler gemacht wird (beispielsweise ist StorageClass falsch), können Sie die Zurückgewinnungsrichtlinie für das PV in wiederherstellen `retain`, Löschen der PVC und PV, und Wiederversuchen des Volumenimportbefehls.

`ontap-nas` Und `ontap-nas-flexgroup` Importe

Jedes Volume wurde mit erstellt `ontap-nas` Treiber ist ein FlexVol auf dem ONTAP Cluster. Importieren von FlexVols mit dem `ontap-nas` Der Treiber funktioniert genauso. Eine FlexVol, die bereits auf einem ONTAP Cluster vorhanden ist, kann als importiert werden `ontap-nas` PVC: Ebenso können FlexGroup Volumes importiert werden als `ontap-nas-flexgroup` VES.

> Ein ONTAP Volume muss vom Typ rw aufweisen, um von Trident zu importieren. Wenn ein Volume vom Typ dp verwendet wird, es ein SnapMirror Ziel-Volume ist. Sie sollten die gespiegelte Beziehung unterbrechen, bevor Sie das Volume in Trident importieren.

> Der `ontap-nas` Der Treiber kann qtrees nicht importieren und verwalten. Der `ontap-nas` Und `ontap-nas-flexgroup` Treiber erlauben keine doppelten Volume-Namen.

Zum Beispiel, um ein Volume mit dem Namen zu importieren `managed_volume` Auf einem Backend mit dem Namen `ontap_nas`, Verwenden Sie den folgenden Befehl:

```
$ tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

So importieren Sie ein Volume mit dem Namen `unmanaged_volume` (Auf dem `ontap_nas backend`), die Trident nicht verwaltet, verwenden Sie den folgenden Befehl:

```
$ tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-file>
--no-manage

+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Bei Verwendung des `--no-manage` Argument: Trident umbenannt oder validiert nicht, ob das Volume angehängt war. Der Volumenimport schlägt fehl, wenn das Volume nicht manuell gemountet wurde.

> Ein zuvor vorhandener Fehler beim Importieren von Volumes mit benutzerdefinierten UnixPermissions wurde behoben. Sie können unixPermissions in Ihrer PVC-Definition oder Back-End-Konfiguration angeben und Astra Trident anweisen, das Volume entsprechend zu importieren.

`ontap-san` Importieren

Astra Trident kann auch ONTAP SAN FlexVols importieren, die eine einzelne LUN enthalten. Dies entspricht dem `ontap-san` Treiber, der für jede PVC und eine LUN innerhalb der FlexVol eine FlexVol erstellt. Sie können das verwenden `tridentctl import` Befehl in gleicher Weise wie in anderen Fällen:

- Geben Sie den Namen des an `ontap-san` Back-End:

- Geben Sie den Namen der zu importierenden FlexVol an. Beachten Sie, dass diese FlexVol nur eine LUN enthält, die importiert werden muss.
- Geben Sie den Pfad der PVC-Definition an, die mit dem verwendet werden muss `-f` Flagge.
- Wählen Sie zwischen PVC-Verwaltung oder -Management. Standardmäßig verwaltet Trident die PVC und benennt die FlexVol und LUN auf dem Back-End um. Um als nicht verwaltetes Volume zu importieren, übergeben Sie den `--no-manage` Flagge.

Beim Importieren eines nicht verwalteten `ontap-san` Volume, Sie sollten sicherstellen, dass die LUN in der FlexVol benannt ist `lun0` Und ist einer Initiatorgruppe mit den gewünschten Initiatoren zugeordnet. Astra Trident übernimmt dies automatisch für einen verwalteten Import.

Astra Trident importiert dann den FlexVol und verknüpft ihn mit der PVC-Definition. Astra Trident ist auch für die FlexVol bekannt `pvc-<uuid>` Formatieren Sie und die LUN innerhalb der FlexVol bis `lun0`.

Es wird empfohlen, Volumes zu importieren, die keine aktiven Verbindungen haben. Wenn Sie ein aktiv verwendetes Volume importieren möchten, klonen Sie zuerst das Volume und führen Sie dann den Import durch.

Beispiel

Um den zu importieren `ontap-san-managed` FlexVol, die auf dem vorhanden ist `ontap_san_default` Back-End, führen Sie das aus `tridentctl import` Befehl als:

```
$ tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+-----------------------------------------+--------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+-----------------------------------------+--------+---------------
+----------+-----------------------------------------+--------+---------+
```

Ein ONTAP-Volume muss vom Typ rw sein, um von Astra Trident importiert werden zu können. Wenn ein Volume vom Typ dp ist, ist es ein SnapMirror Ziel-Volume. Sie sollten die Spiegelbeziehung brechen, bevor Sie das Volume in Astra Trident importieren.

`element` Importieren

Mit Trident können Sie NetApp Element Software/NetApp HCI Volumes in Ihr Kubernetes Cluster importieren. Sie brauchen den Namen Ihres Astra Trident Backend, und den eindeutigen Namen des Volumes und der PVC-Datei als Argumente für die `tridentctl import` Befehl.

```
$ tridentctl import volume element_default element-managed -f pvc-basic-
import.yaml -n trident -d

+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

Der Elementtreiber unterstützt doppelte Volume-Namen. Wenn es doppelte Volume-Namen gibt, gibt Trident Volume Import Prozess einen Fehler zurück. Als Workaround können Sie das Volume klonen und einen eindeutigen Volume-Namen bereitstellen. Importieren Sie dann das geklonte Volume.

`aws-cvs` Importieren

Um ein durch die NetApp Cloud Volumes Service in AWS gesicherte Volume zu importieren, muss das Volume anstelle seines Namens anhand seines Volume-Pfads identifiziert werden.

Um einen zu importieren `aws-cvs` Datenträger auf dem Back-End aufgerufen `awscvs_YEppr` Mit dem Volume-Pfad von `adroit-jolly-swift`, Verwenden Sie den folgenden Befehl:

```
$ tridentctl import volume awscvs_YEppr adroit-jolly-swift -f <path-to-
pvc-file> -n trident

+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | 93 GiB | aws-storage   | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

Der Volume-Pfad ist der Teil des Exportpfads des Volumes nach dem :/. Beispiel: Wenn der Exportpfad lautet `10.0.0.1:/adroit-jolly-swift`, Der Volume-Pfad ist `adroit-jolly-swift`.

`gcp-cvs` Importieren

Importieren von A `gcp-cvs` Das Volume funktioniert genauso wie beim Importieren eines `aws-cvs` Datenmenge:

`azure-netapp-files` Importieren

Um einen zu importieren `azure-netapp-files` Datenträger auf dem Back-End aufgerufen `azurenetappfiles_40517` Mit dem Volume-Pfad `importvol1`, Ausführen des folgenden Befehls:

```
$ tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
|                  NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage   |
file     | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true    |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
```

Der Volume-Pfad für das ANF-Volumen ist im Mount-Pfad nach dem :/ vorhanden. Beispiel: Wenn der Mount-Pfad lautet `10.0.0.2:/importvol1`, Der Volume-Pfad ist `importvol1`.

Bereiten Sie den Knoten „Worker“ vor

Alle Worker-Nodes im Kubernetes Cluster müssen in der Lage sein, die Volumes, die Sie für Ihre Pods bereitgestellt haben, zu mounten. Wenn Sie das verwenden `ontap-nas`, `ontap-nas-economy`, Oder `ontap-nas-flexgroup` Ein Treiber für eines Ihrer Back-Ends werden für Ihre Mitarbeiter-Nodes die NFS-Tools benötigt. Anderenfalls sind iSCSI-Tools erforderlich.

Aktuelle Versionen von RedHat CoreOS haben standardmäßig sowohl NFS als auch iSCSI installiert.

Nach der Installation der NFS- oder iSCSI-Tools sollten Sie die Worker-Nodes immer neu booten, oder das Anbinden von Volumes an Container kann fehlschlagen.

NFS Volumes

Protokoll	Betriebssystem	Befehle
NFS	RHEL/CentOS	`sudo yum install -y nfs-utils`

Protokoll	Betriebssystem	Befehle
NFS	Ubuntu/Debian	`sudo apt-get install -y nfs-common`

Sie sollten sicherstellen, dass der NFS-Dienst während des Startvorgangs gestartet wird.

ISCSI-Volumes

Bei der Verwendung von iSCSI Volumes sollten folgende Punkte berücksichtigt werden:

- Jeder Node im Kubernetes-Cluster muss über einen eindeutigen IQN verfügen. **Dies ist eine notwendige Voraussetzung**.
- Bei Verwendung von RHCOS Version 4.5 oder höher oder RHEL oder CentOS Version 8.2 oder höher mit dem `solidfire-san` Treiber: Stellen Sie sicher, dass der CHAP-Authentifizierungsalgorithmus auf MD5 in gesetzt ist `/etc/iscsi/iscsid.conf`.

```
sudo sed -i 's/^\(node.session.auth.chap_algs\).*/\1 = MD5/'
/etc/iscsi/iscsid.conf
```

- Wenn Sie Worker-Nodes verwenden, die RHEL/RedHat CoreOS mit iSCSI PVS ausführen, stellen Sie sicher, dass die angegeben werden `discard` MountOption in StorageClass für die Inline-Speicherplatzrückgewinnung. Siehe "Die Dokumentation von redhat".

Protokoll	Betriebssystem	Befehle

ISCSI	RHEL/CentOS	1. Installieren Sie die folgenden Systempakete:
		
		`sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-mapper-multipath`
		
		2. Überprüfen Sie, ob die Version von iscsi-Initiator-utils 6.2.0.874-2.el7 oder höher ist:
		
		`rpm -q iscsi-initiator-utils`
		
		3. Scannen auf manuell einstellen:
		
		`sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/' /etc/iscsi/iscsid.conf`
		
		4. Multipathing aktivieren:
		
		`sudo mpathconf --enable --with_multipathd y`
		
		5. Stellen Sie das sicher `iscsid` Und `multipathd` Laufen:
		
		`sudo systemctl enable --now iscsid multipathd`
		
		6. Aktivieren und starten `iscsi`:
		
		`sudo systemctl enable --now iscsi`

Protokoll	Betriebssystem	Befehle

ISCSI	Ubuntu/Debian	1. Installieren Sie die folgenden Systempakete:
		
		`sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools scsitools`
		
		2. Stellen Sie sicher, dass Open-iscsi-Version 2.0.874-5ubuntu2.10 oder höher (für bionic) oder 2.0.874-7.1ubuntu6.1 oder höher (für Brennweite) ist:
		
		`dpkg -l open-iscsi`
		
		3. Scannen auf manuell einstellen:
		
		`sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/' /etc/iscsi/iscsid.conf`
		
		4. Multipathing aktivieren:
		
		`sudo tee /etc/multipath.conf <←'EOF'`
		`defaults {`
		`user_friendly_names yes`
		`find_multipaths yes`
		`}`
		`EOF`
		`sudo systemctl enable --now multipath-tools.service`
		`sudo service multipath-tools restart`
		
		5. Stellen Sie das sicher `open-iscsi` Und `multipath-tools` Sind aktiviert und läuft:
		
		`sudo systemctl status multipath-tools`
		`sudo systemctl enable --now open-iscsi.service`
		`sudo systemctl status open-iscsi`

Für Ubuntu 18.04, müssen Sie Ziel-Ports mit erkennen `iscsiadm` Vor dem Start `open-iscsi` Damit der iSCSI-Daemon gestartet werden kann. Alternativ können Sie den ändern `iscsi` Dienst zu starten `iscsid` Automatisch

Wenn Sie mehr über die automatische Vorbereitung von Workers Node erfahren möchten, die eine Beta-Funktion ist, finden Sie unter "Hier".

Automatische Node-Vorbereitung für Mitarbeiter

Astra Trident kann die erforderlichen automatisch installieren `NFS` Und `iSCSI` Tools auf den Nodes im Kubernetes-Cluster vorhanden. Dies ist ein **Beta Feature** und ist **nicht für** Produktionscluster gedacht. Heute ist die Funktion für Knoten verfügbar, die **CentOS, RHEL und Ubuntu** ausführen.

Diese Funktion enthält Astra Trident ein neues Flag für die Installation: `--enable-node-prep` Bei Installationen mit `tridentctl`. Verwenden Sie bei Implementierungen mit dem Operator Trident die Boolesche Option `enableNodePrep`.

Der `--enable-node-prep` Mit der Installationsoption muss Astra Trident installieren und sicherstellen, dass NFS- und iSCSI-Pakete und/oder -Services ausgeführt werden, wenn ein Volume auf einem Worker-Node angehängt ist. Dies ist eine **Beta-Funktion**, die in Entwicklungs-/Testumgebungen eingesetzt werden soll, die *nicht für den Produktionseinsatz qualifiziert ist.

Wenn der `--enable-node-prep` Flag ist enthalten für die Implementierung von Astra Trident mit `tridentctl`, Hier ist, was passiert:

1. Im Rahmen der Installation registriert Astra Trident die Knoten, auf denen es ausgeführt wird.
2. Wird eine PVC-Anfrage (Persistent Volume Claim) gestellt, erstellt Astra Trident aus einem der von ihm verwalteten Back-Ends ein PV.
3. Wenn Sie PVC in einem POD verwenden, muss Astra Trident das Volume auf dem Node installieren, auf dem der POD läuft. Astra Trident versucht, die erforderlichen NFS/iSCSI-Client-Utilities zu installieren und sicherzustellen, dass die erforderlichen Services aktiv sind. Dies erfolgt, bevor das Volume angehängt wird.

Die Vorbereitung eines Worker-Knotens erfolgt nur einmal im Rahmen des ersten Mounten eines Volumes. Alle nachfolgenden Volume-Mounts sollten erfolgreich sein, solange keine Änderungen außerhalb des Astra Trident erforderlich sind `NFS` Und `iSCSI` Versorgungsunternehmen

Auf diese Weise kann Astra Trident sicherstellen, dass alle Nodes in einem Kubernetes Cluster über die erforderlichen Utilities verfügen, die zum Mounten und Verbinden von Volumes erforderlich sind. Bei NFS-Volumes sollte die Exportrichtlinie auch das Einlegen des Volumes zulassen. Trident kann Exportrichtlinien pro Backend automatisch managen. Alternativ können Benutzer auch Exportrichtlinien out-of-Band managen.

Überwachen Sie Astra Trident

Astra Trident bietet eine Reihe von Prometheus-Kennzahlendpunkten, mit denen Sie die Leistung von Astra Trident überwachen können.

Mit den von Astra Trident bereitgestellten Metriken können Sie:

- Bleiben Sie auf dem Laufenden über den Zustand und die Konfiguration von Astra Trident. Sie können

prüfen, wie erfolgreich Vorgänge sind und ob sie wie erwartet mit den Back-Ends kommunizieren können.

- Untersuchen Sie die Back-End-Nutzungsinformationen und erfahren Sie, wie viele Volumes auf einem Back-End bereitgestellt werden, sowie den belegten Speicherplatz usw.
- Erstellt eine Zuordnung der Anzahl von Volumes, die über verfügbare Back-Ends bereitgestellt werden.
- Verfolgen Sie die Leistung. Sie können sich ansehen, wie lange Astra Trident für die Kommunikation mit Back-Ends und die Durchführung von Vorgängen benötigt.

Die Metriken von Trident sind standardmäßig auf dem Ziel-Port offengelegt `8001` Am `/metrics` endpunkt: Diese Metriken sind bei der Installation von Trident standardmäßig aktiviert.

Was Sie benötigen

- Kubernetes-Cluster mit installiertem Astra Trident
- Eine Prometheus Instanz. Dies kann ein sein "Implementierung von Container-Prometheus" Oder Sie können Prometheus als ein ausführen "Native Applikation".

Schritt 1: Definieren Sie ein Prometheus-Ziel

Sie sollten ein Prometheus Ziel definieren, um die Kennzahlen zu sammeln und Informationen über das Management von Back-Ends Astra Trident, die von ihm erstellten Volumes usw. zu erhalten. Das "Blog" Erläutert, wie Sie mithilfe von Prometheus und Grafana mit Astra Trident Kennzahlen abrufen können. Der Blog erläutert, wie Sie Prometheus als Operator in Ihrem Kubernetes Cluster und die Erstellung eines ServiceMonitor ausführen können, um die Kennzahlen von Astra Trident zu erhalten.

Schritt: Erstellen Sie einen Prometheus ServiceMonitor

Um die Trident Kennzahlen zu verwenden, sollten Sie ein Prometheus ServiceMonitor erstellen, das überwacht `trident-csi` Service und wartet auf den `metrics` Port: Ein Beispiel für ServiceMonitor sieht so aus:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
      release: prom-operator
  spec:
    jobLabel: trident
    selector:
      matchLabels:
        app: controller.csi.trident.netapp.io
    namespaceSelector:
      matchNames:
      - trident
    endpoints:
    - port: metrics
      interval: 15s
```

Diese ServiceMonitor-Definition ruft vom zurückgegebene Kennzahlen ab `trident-csi` Service und insbesondere sucht nach dem `metrics` endpunkt des Dienstes: Das Ergebnis: Prometheus ist jetzt so konfiguriert, dass sie die Kennzahlen von Astra Trident verstehen.

Neben den direkt bei Astra Trident verfügbaren Kennzahlen gibt kubelet auch viele andere Lösungen auf `kubelet_volume_*` Kennzahlen über den Endpunkt der IT-eigenen Kennzahlen. Kubelet kann Informationen über verbundene Volumes bereitstellen und Pods und andere interne Vorgänge, die er übernimmt. Siehe "Hier".

Schritt 3: Abfrage der Trident-Kennzahlen mit PromQL

PromQL ist gut geeignet, um Ausdrücke zu erstellen, die Zeitreihen- oder tabellarische Daten zurückgeben.

Im Folgenden finden Sie einige PromQL-Abfragen, die Sie verwenden können:

Abrufen des Integritätsinformationen zu Trident

- **Prozentsatz der HTTP 2XX-Antworten von Astra Trident**

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Prozentualer Anteil DER REST-Antworten von Astra Trident über Statuscode**

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Durchschnittsdauer in ms der von Astra Trident durchgeführten Operationen**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Holen Sie sich Informationen zur Nutzung von Astra Trident

- **Mittlere Volumengröße**

```
trident_volume_allocated_bytes/trident_volume_count
```

- **Gesamter Volume-Speicherplatz, der von jedem Backend bereitgestellt wird**

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

Individuelle Volume-Nutzung

Dies ist nur aktiviert, wenn auch kubelet-Kennzahlen gesammelt werden.

- **Prozentsatz des verwendeten Speicherplatzes für jedes Volumen**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

AutoSupport Telemetrie von Astra Trident mit Thema

Standardmäßig sendet Astra Trident in einem täglichen Intervall Prometheus-Kennzahlen und grundlegende Backend-Informationen an NetApp.

- Um zu verhindern, dass Astra Trident die Prometheus Kennzahlen und grundlegende Backend-Informationen an NetApp sendet, bestehen Sie am `--silence-autosupport` Fahne während der Installation von Astra Trident.
- Astra Trident kann auch Container-Protokolle per On-Demand an den NetApp Support senden `tridentctl send autosupport`. Sie müssen Astra Trident auslösen, um seine Protokolle hochzuladen. Bevor Sie Protokolle einreichen, sollten Sie die von NetApp akzeptierenhttps://www.netapp.com/company/legal/privacy-policy/["datenschutzrichtlinie"^].
- Sofern nicht angegeben, ruft Astra Trident die Protokolle der letzten 24 Stunden ab.
- Sie können den Zeitrahmen für die Protokollaufbewahrung mit festlegen `--since` Flagge. Beispiel: `tridentctl send autosupport --since=1h`. Diese Informationen werden über ein gesammelt und versendet `trident-autosupport` Container, der zusammen mit Astra Trident installiert wird Sie können das Container-Image unter abrufen "Trident AutoSupport".
- Trident AutoSupport erfasst oder übermittelt keine personenbezogenen Daten oder personenbezogenen Daten. Es kommt mit einem "EULA" Dies gilt nicht für das Trident Container-Image selbst. Weitere Informationen zum Engagement von NetApp für Datensicherheit und Vertrauen "Hier".

Eine von Astra Trident gesendete Beispiellast sieht folgendermaßen aus:

```
{
  "items": [
    {
      "backendUUID": "ff3852e1-18a5-4df4-b2d3-f59f829627ed",
      "protocol": "file",
      "config": {
        "version": 1,
        "storageDriverName": "ontap-nas",
        "debug": false,
        "debugTraceFlags": null,
        "disableDelete": false,
        "serialNumbers": [
          "nwkvzfanek_SN"
        ],
        "limitVolumeSize": ""
      },
      "state": "online",
      "online": true
    }
  ]
}
```

- Die AutoSupport Meldungen werden an den AutoSupport Endpunkt von NetApp gesendet. Wenn Sie zum Speichern von Container-Images eine private Registrierung verwenden, können Sie das verwenden `--image-registry` Flagge.
- Sie können auch Proxy-URLs konfigurieren, indem Sie die Installation YAML-Dateien erstellen. Dies kann mit erfolgen `tridentctl install --generate-custom-yaml` So erstellen Sie die YAML-Dateien und fügen die hinzu `--proxy-url` Argument für das `trident-autosupport` Container in `trident-deployment.yaml`.

Deaktivieren Sie Astra Trident Metriken

Um**-Metriken von der Meldung zu deaktivieren, sollten Sie benutzerdefinierte YAML generieren (mit dem `--generate-custom-yaml` Markieren) und bearbeiten, um die zu entfernen `--metrics` Flagge wird für das aufgerufen `trident-main`Container: