Utilizzare Astra Trident

Astra Trident

NetApp
October 23, 2024

Sommario

Utilizzare Astra Trident

Configurare i backend

Un backend definisce la relazione tra Astra Trident e un sistema storage. Spiega ad Astra Trident come comunicare con quel sistema storage e come Astra Trident dovrebbe eseguire il provisioning dei volumi da esso. Astra Trident offrirà automaticamente pool di storage da backend che insieme soddisfano i requisiti definiti da una classe di storage. Scopri di più sulla configurazione del back-end in base al tipo di sistema storage in uso.

- "Configurare un backend Azure NetApp Files"
- "Configurare un Cloud Volumes Service per il backend AWS"
- "Configurare un Cloud Volumes Service per il backend della piattaforma cloud Google"
- "Configurare un backend NetApp HCI o SolidFire"
- "Configurare un backend con i driver NAS ONTAP"
- "Configurare un backend con i driver SAN ONTAP"
- "Utilizza Astra Trident con Amazon FSX per NetApp ONTAP"

Configurare un backend Azure NetApp Files

Scopri come configurare Azure NetApp Files (ANF) come backend per l'installazione di Astra Trident utilizzando le configurazioni di esempio fornite.

> Il servizio Azure NetApp Files non supporta volumi inferiori a 100 GB. Astra Trident crea automaticamente volumi da 100 GB se viene richiesto un volume più piccolo.

Di cosa hai bisogno

Per configurare e utilizzare un "Azure NetApp Files" back-end, sono necessari i seguenti elementi:

- `subscriptionID` Da un abbonamento Azure con Azure NetApp Files attivato.
- `tenantID`, `clientID`, e. `clientSecret` da un "Registrazione dell'app" In Azure Active Directory con autorizzazioni sufficienti per il servizio Azure NetApp Files. La registrazione dell'applicazione deve utilizzare `Owner` oppure `Contributor` Ruolo predefinito da Azure.

  > Per ulteriori informazioni sui ruoli integrati di Azure, consulta la "Documentazione di Azure".

- Azure `location` che ne contiene almeno uno "subnet delegata".
- Se si utilizza Azure NetApp Files per la prima volta o in una nuova posizione, è necessaria una configurazione iniziale. Vedere "guida rapida".

A proposito di questa attività

In base alla configurazione del backend (subnet, rete virtuale, livello di servizio e posizione), Trident crea volumi ANF su pool di capacità disponibili nella posizione richiesta e corrispondenti al livello di servizio e alla subnet richiesti.

> Astra Trident 21.04.0 e versioni precedenti non supportano i pool di capacità QoS manuali.

Opzioni di configurazione back-end

Per le opzioni di configurazione del backend, consultare la tabella seguente:

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di storage	"azure-netapp-files"
`backendName`	Nome personalizzato o backend dello storage	Nome del driver + "_" + caratteri casuali
`subscriptionID`	L'ID dell'abbonamento dell'abbonamento Azure	
`tenantID`	L'ID tenant di una registrazione app	
`clientID`	L'ID client di una registrazione dell'applicazione	
`clientSecret`	Il segreto del client da una registrazione dell'applicazione	
`serviceLevel`	Uno di `Standard`, `Premium`, o. `Ultra`	"" (casuale)
`location`	Nome della posizione di Azure in cui verranno creati i nuovi volumi	"" (casuale)
`virtualNetwork`	Nome di una rete virtuale con una subnet delegata	"" (casuale)
`subnet`	Nome di una subnet delegata a. `Microsoft.Netapp/volumes`	"" (casuale)
`nfsMountOptions`	Controllo dettagliato delle opzioni di montaggio NFS.	"nfsvers=3"
`limitVolumeSize`	Il provisioning non riesce se le dimensioni del volume richiesto sono superiori a questo valore	"" (non applicato per impostazione predefinita)
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Esempio, `\{"api":false, "method":true}`. Non utilizzare questa opzione a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.	nullo

Modifica di `capacityPools` in un backend esistente, in modo da ridurre il numero di pool di capacità utilizzati per il provisioning, si ottengono volumi orfani, che vengono forniti nel pool/pool di capacità che non fanno parte di `capacityPools` elencare più. Le operazioni di cloning su questi volumi orfani non avranno esito positivo.

Se viene visualizzato l'errore "Nessun pool di capacità trovato" quando si tenta di creare un PVC, è probabile che la registrazione dell'applicazione non abbia le autorizzazioni e le risorse richieste (subnet, rete virtuale, pool di capacità) associate. Astra Trident registrerà le risorse Azure rilevate al momento della creazione del backend quando il debug è attivato. Assicurarsi di controllare se viene utilizzato un ruolo appropriato.

Se si desidera montare i volumi utilizzando NFS versione 4.1, è possibile includere `nfsvers=4` Nell'elenco delle opzioni di montaggio delimitate da virgole, scegliere NFS v4.1. Tutte le opzioni di montaggio impostate in una classe di storage sovrascrivono le opzioni di montaggio impostate in un file di configurazione back-end.

Per impostazione predefinita, è possibile controllare il provisioning di ciascun volume specificando le seguenti opzioni in una sezione speciale del file di configurazione. Vedere gli esempi di configurazione riportati di seguito.

Parametro	Descrizione	Predefinito
`exportRule`	Regola o regole di esportazione per i nuovi volumi	"0.0.0.0/0"
`size`	La dimensione predefinita dei nuovi volumi	"100 G"

Il `exportRule` Il valore deve essere un elenco separato da virgole di qualsiasi combinazione di indirizzi IPv4 o subnet IPv4 nella notazione CIDR.

Per tutti i volumi creati su un backend ANF, Astra Trident copia tutte le etichette presenti su un pool di storage nel volume di storage al momento del provisioning. Gli amministratori dello storage possono definire le etichette per ogni pool di storage e raggruppare tutti i volumi creati in un pool di storage. In questo modo è possibile differenziare i volumi in base a una serie di etichette personalizzabili fornite nella configurazione di back-end.

Esempio 1: Configurazione minima

Questa è la configurazione backend minima assoluta. Con questa configurazione, Astra Trident scopre tutti gli account NetApp, i pool di capacità e le subnet delegate ad ANF in ogni sede in tutto il mondo e colloca nuovi volumi su uno di essi in modo casuale.

Questa configurazione è ideale quando si inizia a utilizzare ANF e si provano le cose, ma in pratica si desidera fornire un ambito aggiuntivo per i volumi che si esegue il provisioning.

```
{
    "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"
}
```

Esempio 2: Singola posizione e configurazione specifica del livello di servizio

Questa configurazione di back-end consente di posizionare i volumi in Azure `eastus` posizione in a. `Premium` pool di capacità. Astra Trident rileva automaticamente tutte le subnet delegate ad ANF in quella posizione e inserisce un nuovo volume su una di esse in modo casuale.

```
{
    "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"
}
```

Esempio 3: Configurazione avanzata

Questa configurazione di back-end riduce ulteriormente l'ambito del posizionamento del volume in una singola subnet e modifica alcune impostazioni predefinite di provisioning del volume.

```
{
    "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"
    }
}
```

Esempio 4: Configurazione del pool di storage virtuale

Questa configurazione di back-end definisce più pool di storage in un singolo file. Ciò è utile quando si dispone di più pool di capacità che supportano diversi livelli di servizio e si desidera creare classi di storage in Kubernetes che ne rappresentano.

```
{
    "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",
        }
    ]
}
```

Quanto segue `StorageClass` le definizioni si riferiscono ai pool di storage sopra indicati. Utilizzando `parameters.selector` è possibile specificare per ciascun campo `StorageClass` il pool di visualizzazioni utilizzato per ospitare un volume. Gli aspetti del volume saranno definiti nel pool selezionato.

```
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
```

Quali sono le prossime novità?

Dopo aver creato il file di configurazione back-end, eseguire il seguente comando:

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

Se la creazione del backend non riesce, si è verificato un errore nella configurazione del backend. È possibile visualizzare i log per determinare la causa eseguendo il seguente comando:

```
tridentctl logs
```

Dopo aver identificato e corretto il problema con il file di configurazione, è possibile eseguire nuovamente il comando create.

Configurare un CVS per il backend AWS

Scopri come configurare NetApp Cloud Volumes Service (CVS) per AWS come backend per la tua

installazione Astra Trident utilizzando le configurazioni di esempio fornite.

> Cloud Volumes Service per AWS non supporta volumi inferiori a 100 GB. Trident crea automaticamente volumi da 100 GB se viene richiesto un volume più piccolo.

Di cosa hai bisogno

Per configurare e utilizzare "Cloud Volumes Service per AWS" back-end, sono necessari i seguenti elementi:

- Un account AWS configurato con NetApp CVS
- Regione API, URL e chiavi per il tuo account CVS

Opzioni di configurazione back-end

Per le opzioni di configurazione del backend, consultare la tabella seguente:

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di storage	"aws-cvs"
`backendName`	Nome personalizzato o backend dello storage	Nome del driver + "_" + parte della chiave API
`apiRegion`	Regione dell'account CVS. È possibile trovare il valore nel portale web CVS in account settings/API access (Impostazioni account/accesso API).	
`apiURL`	URL API DELL'ACCOUNT CVS. È possibile trovare il valore nel portale web CVS in account settings/API access (Impostazioni account/accesso API).	
`apiKey`	Chiave API dell'account CVS. È possibile trovare il valore nel portale web CVS in account settings/API access (Impostazioni account/accesso API).	
`secretKey`	Chiave segreta dell'account CVS. È possibile trovare il valore nel portale web CVS in account settings/API access (Impostazioni account/accesso API).	

Parametro	Descrizione	Predefinito
`proxyURL`	URL del proxy se il server proxy ha richiesto di connettersi all'account CVS. Il server proxy può essere un proxy HTTP o un proxy HTTPS. Per un proxy HTTPS, la convalida del certificato viene ignorata per consentire l'utilizzo di certificati autofirmati nel server proxy. I server proxy con autenticazione abilitata non sono supportati.	
`nfsMountOptions`	Controllo dettagliato delle opzioni di montaggio NFS.	"nfsvers=3"
`limitVolumeSize`	Il provisioning non riesce se le dimensioni del volume richiesto sono superiori a questo valore	"" (non applicato per impostazione predefinita)
`serviceLevel`	Il livello di servizio CVS per i nuovi volumi. I valori sono "standard", "premium" e "Extreme".	"standard"
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Esempio, `\{"api":false, "method":true}`. Non utilizzare questa opzione a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.	nullo

> `apiURL` è unico per ciascuno `apiRegion`. Ad esempio, US-West-2 `apiRegion` ha il https://cv.us-west-2.netapp.com:8080/v1/ `apiURL`. Allo stesso modo, gli Stati Uniti-est-1 `apiRegion` ha il https://cds-aws-bundles.netapp.com:8080/v1/ `apiURL`. Verificare che il dashboard CVS sia corretto `apiRegion` e. `apiURL` parametri per la configurazione back-end.

Ogni backend esegue il provisioning dei volumi in una singola regione AWS. Per creare volumi in altre regioni, è possibile definire backend aggiuntivi.

Per impostazione predefinita, è possibile controllare il provisioning di ciascun volume specificando le seguenti opzioni in una sezione speciale del file di configurazione. Vedere gli esempi di configurazione riportati di seguito.

Parametro	Descrizione	Predefinito
`exportRule`	Regola o regole di esportazione per i nuovi volumi	"0.0.0.0/0"
`snapshotDir`	Controlla la visibilità di `.snapshot` directory	"falso"
`snapshotReserve`	Percentuale di volume riservato agli snapshot	"" (accettare CVS come valore predefinito 0)

Parametro	Descrizione	Predefinito
`size`	Le dimensioni dei nuovi volumi	"100 G"

Il `exportRule` Il valore deve essere un elenco separato da virgole di qualsiasi combinazione di indirizzi IPv4 o subnet IPv4 nella notazione CIDR.

> Per tutti i volumi creati su un backend CVS AWS, Astra Trident copia tutte le etichette presenti su un pool di storage nel volume di storage al momento del provisioning. Gli amministratori dello storage possono definire le etichette per ogni pool di storage e raggruppare tutti i volumi creati in un pool di storage. In questo modo è possibile differenziare i volumi in base a una serie di etichette personalizzabili fornite nella configurazione di back-end.

Esempio 1: Configurazione minima

Questa è la configurazione backend minima assoluta.

Questa configurazione è ideale quando si inizia a utilizzare CVS AWS e si provano le cose, ma in pratica si desidera fornire un ambito aggiuntivo per i volumi che si esegue il provisioning.

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

Esempio 2: Configurazione a livello di servizio singolo

Questo esempio mostra un file backend che applica gli stessi aspetti a tutti gli storage creati da Astra Trident nella regione AWS us-East-1. Questo esempio mostra anche l'utilizzo di `proxyURL` nel file backend.

```
{
    "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"
    }
}
```

Esempio 3: Configurazione del pool di storage virtuale

Questo esempio mostra il file di definizione back-end configurato con i pool di storage virtuali insieme a StorageClasses che fanno riferimento a questi.

Nel file di definizione del backend di esempio mostrato di seguito, vengono impostati valori predefiniti specifici per tutti i pool di storage, che impostano `snapshotReserve` al 5% e a. `exportRule` a 0.0.0.0/0. I pool di storage virtuali sono definiti in `storage` sezione. In questo esempio, ogni singolo pool di storage imposta il proprio `serviceLevel`e alcuni pool sovrascrivono i valori predefiniti.

```
{
    "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"
        }
    ]
}
```

Le seguenti definizioni di StorageClass si riferiscono ai pool di storage di cui sopra. Utilizzando `parameters.selector` È possibile specificare per ogni StorageClass il pool virtuale utilizzato per ospitare un volume. Gli aspetti del volume saranno definiti nel pool selezionato.

Il primo StorageClass (`cvs-extreme-extra-protection`) viene mappato al primo pool di storage virtuale. Questo è l'unico pool che offre performance estreme con una riserva di snapshot del 10%. L'ultima StorageClass (`cvs-extra-protection`) richiama qualsiasi pool di storage che fornisce una riserva di snapshot del 10%. Astra Trident decide quale pool di storage virtuale è selezionato e garantisce che il requisito di riserva snapshot sia soddisfatto.

```
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
```

Quali sono le prossime novità?

Dopo aver creato il file di configurazione back-end, eseguire il seguente comando:

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

Se la creazione del backend non riesce, si è verificato un errore nella configurazione del backend. È possibile visualizzare i log per determinare la causa eseguendo il seguente comando:

```
tridentctl logs
```

Dopo aver identificato e corretto il problema con il file di configurazione, è possibile eseguire nuovamente il comando create.

Configurare un CVS per il backend GCP

Scopri come configurare NetApp Cloud Volumes Service (CVS) per la piattaforma cloud Google (GCP) come back-end per l'installazione di Astra Trident utilizzando le configurazioni di esempio fornite.

> NetApp Cloud Volumes Service per Google Cloud non supporta volumi CVS-Performance di dimensioni inferiori a 100 GiB o volumi CVS di dimensioni inferiori a 300 GiB. Astra Trident crea automaticamente volumi di dimensioni minime se il volume richiesto è inferiore alla dimensione minima.

Di cosa hai bisogno

Per configurare e utilizzare "Cloud Volumes Service per Google Cloud" back-end, sono necessari i seguenti elementi:

- Un account Google Cloud configurato con NetApp CVS
- Numero di progetto dell'account Google Cloud
- Account di servizio Google Cloud con `netappcloudvolumes.admin` ruolo
- File delle chiavi API per l'account del servizio CVS

Astra Trident ora include il supporto per volumi più piccoli con l'impostazione predefinita "tipi di servizio":[tipo di servizio CVS su GCP^]. Per i backend creati con `storageClass=software`, I volumi avranno ora una dimensione di provisioning minima di 300 GiB. CVS attualmente fornisce questa funzione in disponibilità controllata e non fornisce supporto tecnico. Gli utenti devono iscriversi per accedere ai volumi inferiori a 1 TiB "qui". NetApp consiglia ai clienti di consumare volumi inferiori a 1 TiB per carichi di lavoro **non in produzione**.

> Quando si implementano backend utilizzando il tipo di servizio CVS predefinito (`storageClass=software`), gli utenti devono ottenere l'accesso alla funzione volumi sub-1TiB su GCP per i numeri di progetto e gli ID progetto in questione. Questo è necessario per Astra Trident per eseguire il provisioning di volumi inferiori a 1 TiB. In caso contrario, le creazioni di volumi non verranno superate per PVC inferiori a 600 GiB. Ottenere l'accesso a volumi inferiori a 1 TiB utilizzando "questo modulo".

I volumi creati da Astra Trident per il livello di servizio CVS predefinito verranno forniti come segue:

- I PVC di dimensioni inferiori a 300 GiB determinano la creazione di un volume CVS da 300 GiB da parte di Astra Trident.
- I PVC compresi tra 300 GiB e 600 GiB determineranno la creazione di un volume CVS della dimensione richiesta da parte di Astra Trident.
- I PVC compresi tra 600 GiB e 1 TIB determineranno la creazione di un volume CVS 1TiB da parte di Astra Trident.
- I PVC superiori a 1 TIB determineranno la creazione da parte di Astra Trident di un volume CVS della dimensione richiesta.

Opzioni di configurazione back-end

Per le opzioni di configurazione del backend, consultare la tabella seguente:

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di storage	"gcp-cvs"
`backendName`	Nome personalizzato o backend dello storage	Nome del driver + "_" + parte della chiave API

Parametro	Descrizione	Predefinito
`storageClass`	Tipo di storage. Scegliere tra `hardware` (prestazioni ottimizzate) o. `software` (Tipo di servizio CVS)	
`projectNumber`	Numero di progetto dell'account Google Cloud. Il valore si trova nella home page del portale Google Cloud.	
`apiRegion`	Regione dell'account CVS. È la regione in cui il back-end eseguirà il provisioning dei volumi.	
`apiKey`	Chiave API per l'account del servizio Google Cloud con `netappcloudvolumes.admin` ruolo. Include il contenuto in formato JSON di un file di chiave privata dell'account di un servizio Google Cloud (copia integrale nel file di configurazione del backend).	
`proxyURL`	URL del proxy se il server proxy ha richiesto di connettersi all'account CVS. Il server proxy può essere un proxy HTTP o un proxy HTTPS. Per un proxy HTTPS, la convalida del certificato viene ignorata per consentire l'utilizzo di certificati autofirmati nel server proxy. I server proxy con autenticazione abilitata non sono supportati.	
`nfsMountOptions`	Controllo dettagliato delle opzioni di montaggio NFS.	"nfsvers=3"
`limitVolumeSize`	Il provisioning non riesce se le dimensioni del volume richiesto sono superiori a questo valore	"" (non applicato per impostazione predefinita)
`serviceLevel`	Il livello di servizio CVS per i nuovi volumi. I valori sono "standard", "premium" e "Extreme".	"standard"
`network`	Rete GCP utilizzata per volumi CVS	"predefinito"
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Esempio, `\{"api":false, "method":true}`. Non utilizzare questa opzione a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.	nullo

Se si utilizza una rete VPC condivisa, entrambi `projectNumber` e. `hostProjectNumber` deve essere specificato. In tal caso, `projectNumber` è il progetto di servizio, e. `hostProjectNumber` è il progetto host.

Il `apiRegion` Rappresenta la regione GCP in cui Astra Trident crea volumi CVS. Astra Trident può montare e collegare volumi su nodi Kubernetes che appartengono alla stessa regione GCP. Quando si crea un backend, è importante assicurarsi `apiRegion` Corrisponde alla regione in cui vengono implementati i nodi Kubernetes. Durante la creazione di cluster Kubernetes a più aree, i volumi CVS creati in un dato `apiRegion` Può essere utilizzato solo nei carichi di lavoro pianificati su nodi che si trovano nella stessa regione GCP.

> `storageClass` è un parametro facoltativo che è possibile utilizzare per selezionare il desiderato "Tipo di servizio CVS". È possibile scegliere tra il tipo di servizio CVS di base (`storageClass=software`) O il tipo di servizio CVS-Performance (`storageClass=hardware`), che Trident utilizza per impostazione predefinita. Assicurarsi di specificare un `apiRegion` Che fornisce il rispettivo CVS `storageClass` nella definizione di back-end.

> L'integrazione di Astra Trident con il tipo di servizio CVS di base su Google Cloud è una funzionalità **beta**, non destinata ai carichi di lavoro di produzione. Trident è **completamente supportato** con il tipo di servizio CVS-Performance e lo utilizza per impostazione predefinita.

Ogni back-end esegue il provisioning dei volumi in una singola area di Google Cloud. Per creare volumi in altre regioni, è possibile definire backend aggiuntivi.

Per impostazione predefinita, è possibile controllare il provisioning di ciascun volume specificando le seguenti opzioni in una sezione speciale del file di configurazione. Vedere gli esempi di configurazione riportati di seguito.

Parametro	Descrizione	Predefinito
`exportRule`	Regola o regole di esportazione per i nuovi volumi	"0.0.0.0/0"
`snapshotDir`	Accesso a `.snapshot` directory	"falso"
`snapshotReserve`	Percentuale di volume riservato agli snapshot	"" (accettare CVS come valore predefinito 0)
`size`	Le dimensioni dei nuovi volumi	"100 Gi"

Il `exportRule` Il valore deve essere un elenco separato da virgole di qualsiasi combinazione di indirizzi IPv4 o subnet IPv4 nella notazione CIDR.

> Per tutti i volumi creati su un backend CVS Google Cloud, Trident copia tutte le etichette presenti su un pool di storage nel volume di storage al momento del provisioning. Gli amministratori dello storage possono definire le etichette per ogni pool di storage e raggruppare tutti i volumi creati in un pool di storage. In questo modo è possibile differenziare i volumi in base a una serie di etichette personalizzabili fornite nella configurazione di back-end.

Esempio 1: Configurazione minima

Questa è la configurazione backend minima assoluta.

```
{
    "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-----
        <key_value>
        -----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"
    }
}
```

Esempio 2: Configurazione del tipo di servizio CVS di base

Questo esempio mostra una definizione di back-end che utilizza il tipo di servizio CVS di base, che è destinato ai carichi di lavoro generici e fornisce performance leggere/moderate, insieme ad un'elevata disponibilità zonale.

```
{
    "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-----
        <key_value>
        -----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"
    }
}
```

Esempio 3: Configurazione a livello di servizio singolo

Questo esempio mostra un file backend che applica gli stessi aspetti a tutti gli storage creati da Astra Trident nella regione di Google Cloud us-west2. Questo esempio mostra anche l'utilizzo di `proxyURL` nel file di configurazione back-end.

```
{
    "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-----
        <key_value>
        -----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"
    }
}
```

Esempio 4: Configurazione del pool di storage virtuale

Questo esempio mostra il file di definizione back-end configurato con i pool di storage virtuali insieme a. `StorageClasses` che fanno riferimento a loro.

Nel file di definizione del backend di esempio mostrato di seguito, vengono impostati valori predefiniti specifici per tutti i pool di storage, che impostano `snapshotReserve` al 5% e a. `exportRule` a 0.0.0.0/0. I pool di storage virtuali sono definiti in `storage` sezione. In questo esempio, ogni singolo pool di storage imposta il proprio `serviceLevel`e alcuni pool sovrascrivono i valori predefiniti.

```
{
    "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-----
        <key_value>
        -----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"
        }
    ]
}
```

Le seguenti definizioni di StorageClass si riferiscono ai pool di storage di cui sopra. Utilizzando `parameters.selector` È possibile specificare per ogni StorageClass il pool virtuale utilizzato per ospitare un volume. Gli aspetti del volume saranno definiti nel pool selezionato.

Il primo StorageClass (`cvs-extreme-extra-protection`) viene mappato al primo pool di storage virtuale. Questo è l'unico pool che offre performance estreme con una riserva di snapshot del 10%. L'ultima StorageClass (`cvs-extra-protection`) richiama qualsiasi pool di storage che fornisce una riserva di snapshot del 10%. Astra Trident decide quale pool di storage virtuale è selezionato e garantisce che il requisito

di riserva snapshot sia soddisfatto.

```
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
```

Quali sono le prossime novità?

Dopo aver creato il file di configurazione back-end, eseguire il seguente comando:

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

Se la creazione del backend non riesce, si è verificato un errore nella configurazione del backend. È possibile visualizzare i log per determinare la causa eseguendo il seguente comando:

```
tridentctl logs
```

Dopo aver identificato e corretto il problema con il file di configurazione, è possibile eseguire nuovamente il comando create.

Configurare un backend NetApp HCI o SolidFire

Scopri come creare e utilizzare un backend Element con l'installazione di Astra Trident.

Di cosa hai bisogno

- Un sistema storage supportato che esegue il software Element.
- Credenziali per un amministratore del cluster NetApp HCI/SolidFire o un utente tenant in grado di gestire i volumi.
- Tutti i nodi di lavoro di Kubernetes devono disporre dei tool iSCSI appropriati. Vedere "informazioni sulla preparazione del nodo di lavoro".

Cosa devi sapere

Il `solidfire-san` il driver di storage supporta entrambe le modalità di volume: file e block. Per `Filesystem` VolumeMode, Astra Trident crea un volume e un filesystem. Il tipo di file system viene specificato da StorageClass.

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
`solidfire-san`	ISCSI	Blocco	RWO, ROX, RWX	Nessun filesystem. Dispositivo a blocchi raw.
`solidfire-san`	ISCSI	Blocco	RWO, ROX, RWX	Nessun filesystem. Dispositivo a blocchi raw.
`solidfire-san`	ISCSI	Filesystem	RWO, ROX	`xfs`, `ext3`, `ext4`
`solidfire-san`	ISCSI	Filesystem	RWO, ROX	`xfs`, `ext3`, `ext4`

Astra Trident utilizza CHAP quando funziona come provider CSI avanzato. Se si utilizza CHAP (che è l'impostazione predefinita per CSI), non sono necessarie ulteriori operazioni di preparazione. Si consiglia di impostare in modo esplicito `UseCHAP` Opzione per utilizzare CHAP con Trident non CSI. In caso contrario, vedere "qui".

I gruppi di accesso ai volumi sono supportati solo dal framework convenzionale non CSI per Astra Trident. Se configurato per funzionare in modalità CSI, Astra Trident utilizza CHAP.

In caso contrario `AccessGroups` oppure `UseCHAP` viene impostata una delle seguenti regole:

- Se l'impostazione predefinita `trident` viene rilevato un gruppo di accesso, vengono utilizzati i gruppi di accesso.
- Se non viene rilevato alcun gruppo di accesso e Kubernetes versione 1.7 o successiva, viene utilizzato CHAP.

Opzioni di configurazione back-end

Per le opzioni di configurazione del backend, consultare la tabella seguente:

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di storage	"Solidfire-san"
`backendName`	Nome personalizzato o backend dello storage	"SolidFire_" + indirizzo IP dello storage (iSCSI)
`Endpoint`	MVIP per il cluster SolidFire con credenziali tenant	
`SVIP`	Porta e indirizzo IP dello storage (iSCSI)	
`labels`	Set di etichette arbitrarie formattate con JSON da applicare sui volumi.	""

Parametro	Descrizione	Predefinito
`TenantName`	Nome tenant da utilizzare (creato se non trovato)	
`InitiatorIFace`	Limitare il traffico iSCSI a un'interfaccia host specifica	"predefinito"
`UseCHAP`	Utilizzare CHAP per autenticare iSCSI	vero
`AccessGroups`	Elenco degli ID del gruppo di accesso da utilizzare	Trova l'ID di un gruppo di accesso denominato "tridente"
`Types`	Specifiche QoS	
`limitVolumeSize`	Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore	"" (non applicato per impostazione predefinita)
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Ad esempio, {"api":false,} method":true	nullo

Non utilizzare `debugTraceFlags` a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.

Per tutti i volumi creati, Astra Trident copia tutte le etichette presenti in un pool di storage nel LUN dello storage di backup al momento del provisioning. Gli amministratori dello storage possono definire le etichette per ogni pool di storage e raggruppare tutti i volumi creati in un pool di storage. In questo modo è possibile differenziare i volumi in base a una serie di etichette personalizzabili fornite nella configurazione di back-end.

Esempio 1: Configurazione back-end per `solidfire-san` driver con tre tipi di volume

Questo esempio mostra un file backend che utilizza l'autenticazione CHAP e modellazione di tre tipi di volume con specifiche garanzie di QoS. È molto probabile che si definiscano le classi di storage per utilizzarle utilizzando `IOPS` parametro della classe di storage.

```
{
    "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}}]
}
```

Esempio 2: Configurazione del backend e della classe di storage per `solidfire-san` driver con pool di storage virtuali

Questo esempio mostra il file di definizione back-end configurato con i pool di storage virtuali insieme a StorageClasses che fanno riferimento a questi.

Nel file di definizione del backend di esempio mostrato di seguito, vengono impostati valori predefiniti specifici per tutti i pool di storage, che impostano `type` In Silver. I pool di storage virtuali sono definiti in `storage` sezione. In questo esempio, alcuni pool di storage impostano il proprio tipo e alcuni pool sovrascrivono i valori predefiniti precedentemente impostati.

```
{
    "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"
        }
    ]
}
```

Le seguenti definizioni di StorageClass si riferiscono ai pool di storage virtuali sopra indicati. Utilizzando il `parameters.selector` Ciascun StorageClass richiama i pool virtuali che possono essere utilizzati per ospitare un volume. Gli aspetti del volume saranno definiti nel pool virtuale scelto.

Il primo StorageClass (`solidfire-gold-four`) verrà mappato al primo pool di storage virtuale. Questo è

l'unico pool che offre performance eccellenti con un `Volume Type QoS` Dell'oro. L'ultima StorageClass (`solidfire-silver`) definisce qualsiasi pool di storage che offra performance di livello silver. Astra Trident deciderà quale pool di storage virtuale è selezionato e garantirà il rispetto dei requisiti di storage.

```
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"
```

Trova ulteriori informazioni

- "Gruppi di accesso ai volumi"

Configurare un backend con i driver SAN ONTAP

Scopri come configurare un backend ONTAP con i driver SAN ONTAP.

- "Preparazione"
- "Configurazione ed esempi"

Autorizzazioni utente

Astra Trident prevede di essere eseguito come amministratore di ONTAP o SVM, in genere utilizzando `admin` utente del cluster o un `vsadmin` Utente SVM o un utente con un nome diverso che ha lo stesso ruolo. Per le implementazioni di Amazon FSX per NetApp ONTAP, Astra Trident prevede di essere eseguito come amministratore di ONTAP o SVM, utilizzando il cluster `fsxadmin` utente o a. `vsadmin` Utente SVM o un utente con un nome diverso che ha lo stesso ruolo. Il `fsxadmin` user è un sostituto limitato per l'utente amministratore del cluster.

> Se si utilizza `limitAggregateUsage` parametro, sono richieste le autorizzazioni di amministrazione del cluster. Quando si utilizza Amazon FSX per NetApp ONTAP con Astra Trident, il `limitAggregateUsage` il parametro non funziona con `vsadmin` e. `fsxadmin` account utente. L'operazione di configurazione non riesce se si specifica questo parametro.

Preparazione

Scopri come preparare la configurazione di un backend ONTAP con i driver SAN ONTAP. Per tutti i backend ONTAP, Astra Trident richiede almeno un aggregato assegnato alla SVM.

È inoltre possibile eseguire più di un driver e creare classi di storage che puntino all'una o all'altra. Ad esempio, è possibile configurare un `san-dev` classe che utilizza `ontap-san` driver e a. `san-default` classe che utilizza `ontap-san-economy` uno.

Tutti i nodi di lavoro di Kubernetes devono disporre dei tool iSCSI appropriati. Vedere "qui" per ulteriori dettagli.

Autenticazione

Astra Trident offre due modalità di autenticazione di un backend ONTAP.

- Basato sulle credenziali: Nome utente e password di un utente ONTAP con le autorizzazioni richieste. Si consiglia di utilizzare un ruolo di accesso di sicurezza predefinito, ad esempio `admin` oppure `vsadmin` Per garantire la massima compatibilità con le versioni di ONTAP.
- Basato su certificato: Astra Trident può anche comunicare con un cluster ONTAP utilizzando un certificato installato sul backend. In questo caso, la definizione di backend deve contenere i valori codificati in Base64 del certificato client, della chiave e del certificato CA attendibile, se utilizzato (consigliato).

Gli utenti possono anche scegliere di aggiornare i back-end esistenti, scegliendo di passare da basato su credenziali a basato su certificato e viceversa. Se **vengono forniti sia credenziali che certificati**, Astra Trident utilizza per impostazione predefinita i certificati mentre emette un avviso per rimuovere le credenziali dalla definizione di backend.

Abilitare l'autenticazione basata su credenziali

Astra Trident richiede le credenziali di un amministratore con ambito SVM/cluster per comunicare con il backend ONTAP. Si consiglia di utilizzare ruoli standard predefiniti, ad esempio `admin` oppure `vsadmin`. Ciò garantisce la compatibilità con le future release di ONTAP che potrebbero esporre le API delle funzionalità da utilizzare nelle future release di Astra Trident. È possibile creare e utilizzare un ruolo di accesso di sicurezza personalizzato con Astra Trident, ma non è consigliato.

Una definizione di back-end di esempio avrà un aspetto simile al seguente:

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

Tenere presente che la definizione di backend è l'unica posizione in cui le credenziali vengono memorizzate in testo normale. Una volta creato il backend, i nomi utente e le password vengono codificati con Base64 e memorizzati come segreti Kubernetes. La creazione/l'updation di un backend è l'unico passaggio che richiede la conoscenza delle credenziali. Pertanto, si tratta di un'operazione di sola amministrazione, che deve essere eseguita dall'amministratore Kubernetes/storage.

Abilitare l'autenticazione basata su certificato

I backend nuovi ed esistenti possono utilizzare un certificato e comunicare con il backend ONTAP. Nella definizione di backend sono necessari tre parametri.

- ClientCertificate: Valore del certificato client codificato con base64.
- ClientPrivateKey: Valore codificato in base64 della chiave privata associata.
- TrustedCACertificate: Valore codificato in base64 del certificato CA attendibile. Se si utilizza una CA attendibile, è necessario fornire questo parametro. Questa operazione può essere ignorata se non viene utilizzata alcuna CA attendibile.

Un workflow tipico prevede i seguenti passaggi.

Fasi

1. Generare un certificato e una chiave del client. Durante la generazione, impostare il nome comune (CN) sull'utente ONTAP per l'autenticazione come.

   ```
   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. Aggiungere un certificato CA attendibile al cluster ONTAP. Questo potrebbe essere già gestito

dall'amministratore dello storage. Ignorare se non viene utilizzata alcuna CA attendibile.

```
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. Installare il certificato e la chiave del client (dal passaggio 1) sul cluster ONTAP.

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

4. Verificare che il ruolo di accesso di sicurezza di ONTAP supporti `cert` metodo di autenticazione.

```
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. Verifica dell'autenticazione utilizzando il certificato generato. Sostituire <LIF di gestione ONTAP> e <vserver name> con IP LIF di gestione e nome SVM.

```
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. Codifica certificato, chiave e certificato CA attendibile con Base64.

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

7. Creare il backend utilizzando i valori ottenuti dal passaggio precedente.

```
$ 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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Aggiornare i metodi di autenticazione o ruotare le credenziali

È possibile aggiornare un backend esistente per utilizzare un metodo di autenticazione diverso o per ruotare le credenziali. Questo funziona in entrambi i modi: I backend che utilizzano il nome utente/la password possono essere aggiornati per utilizzare i certificati; i backend che utilizzano i certificati possono essere aggiornati in base al nome utente/alla password. A tale scopo, utilizzare un aggiornamento `backend.json` file contenente i parametri da eseguire `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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

> Quando si ruotano le password, l'amministratore dello storage deve prima aggiornare la password per l'utente su ONTAP. Seguito da un aggiornamento back-end. Durante la rotazione dei certificati, è possibile aggiungere più certificati all'utente. Il backend viene quindi aggiornato per utilizzare il nuovo certificato, dopodiché il vecchio certificato può essere cancellato dal cluster ONTAP.

L'aggiornamento di un backend non interrompe l'accesso ai volumi già creati, né influisce sulle connessioni dei volumi effettuate successivamente. Un aggiornamento back-end corretto indica che Astra Trident può comunicare con il backend ONTAP e gestire le future operazioni sui volumi.

Specifica igroups

Astra Trident utilizza igroups per controllare l'accesso ai volumi (LUN) forniti. Gli amministratori hanno due opzioni per specificare igroups per i backend:

- Astra Trident può creare e gestire automaticamente un igroup per backend. Se `igroupName` Non è incluso nella definizione di backend, Astra Trident crea un igroup denominato `trident-<backend-UUID>` Su SVM. In questo modo, ciascun backend disporrà di un igroup dedicato e gestirà l'aggiunta/eliminazione automatica degli IQN dei nodi Kubernetes.
- In alternativa, gli igroups pre-creati possono essere forniti anche in una definizione di back-end. Questa

operazione può essere eseguita utilizzando `igroupName` parametro di configurazione. Astra Trident aggiungerà/eliminerà gli IQN dei nodi Kubernetes all'igroup preesistente.

Per i backend che hanno `igroupName` definito, il `igroupName` può essere eliminato con un `tridentctl backend update` Per fare in modo che Astra Trident gestisca automaticamente igroups. In questo modo, l'accesso ai volumi già collegati ai carichi di lavoro non verrà disturbato. Le connessioni future verranno gestite utilizzando il igroup Astra Trident creato.

> Dedicare un igroup per ogni istanza unica di Astra Trident è una Best practice che è vantaggiosa per l'amministratore Kubernetes e per l'amministratore dello storage. CSI Trident automatizza l'aggiunta e la rimozione degli IQN dei nodi del cluster all'igroup, semplificando notevolmente la gestione. Quando si utilizza la stessa SVM in ambienti Kubernetes (e installazioni Astra Trident), l'utilizzo di un igroup dedicato garantisce che le modifiche apportate a un cluster Kubernetes non influiscano sugli igroups associati a un altro. Inoltre, è importante garantire che ciascun nodo del cluster Kubernetes disponga di un IQN univoco. Come indicato in precedenza, Astra Trident gestisce automaticamente l'aggiunta e la rimozione di IQN. Il riutilizzo degli IQN tra gli host può portare a scenari indesiderati in cui gli host si scambiano e l'accesso alle LUN viene negato.

Se Astra Trident è configurato per funzionare come provider CSI, gli IQN dei nodi Kubernetes vengono aggiunti/rimossi automaticamente dall'igroup. Quando i nodi vengono aggiunti a un cluster Kubernetes, `trident-csi` DemonSet implementa un pod (`trident-csi-xxxxx`) sui nodi appena aggiunti e registra i nuovi nodi a cui è possibile collegare i volumi. Gli IQN dei nodi vengono aggiunti anche all'igroup del backend. Un insieme simile di passaggi gestisce la rimozione degli IQN quando i nodi vengono cordonati, scaricati e cancellati da Kubernetes.

Se Astra Trident non viene eseguito come CSI Provisioner, l'igroup deve essere aggiornato manualmente per contenere gli IQN iSCSI di ogni nodo di lavoro nel cluster Kubernetes. Gli IQN dei nodi che fanno parte del cluster Kubernetes dovranno essere aggiunti all'igroup. Analogamente, gli IQN dei nodi rimossi dal cluster Kubernetes devono essere rimossi dall'igroup.

Autenticare le connessioni con CHAP bidirezionale

Astra Trident può autenticare le sessioni iSCSI con CHAP bidirezionale per `ontap-san` e. `ontap-san-economy` driver. Per eseguire questa operazione, è necessario attivare `useCHAP` nella definizione del backend. Quando è impostato su `true`, Astra Trident configura la protezione predefinita dell'iniziatore SVM su CHAP bidirezionale e imposta il nome utente e i segreti del file backend. NetApp consiglia di utilizzare CHAP bidirezionale per autenticare le connessioni. Vedere la seguente configurazione di esempio:

```
{
    "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",
}
```

> Il `useCHAP` Parameter è un'opzione booleana che può essere configurata una sola volta. L'impostazione predefinita è false. Una volta impostato su true, non è possibile impostarlo su false.

Oltre a `useCHAP=true`, il `chapInitiatorSecret`, `chapTargetInitiatorSecret`, `chapTargetUsername`, e. `chapUsername` i campi devono essere inclusi nella definizione di backend. I segreti possono essere modificati dopo la creazione di un backend mediante l'esecuzione `tridentctl update`.

Come funziona

Per impostazione `useCHAP` A vero, l'amministratore dello storage istruisce Astra Trident a configurare CHAP sul backend dello storage. Ciò include quanto segue:

- Impostazione di CHAP su SVM:
  - Se il tipo di protezione initiator predefinito di SVM è None (impostato per impostazione predefinita) **e** non sono presenti LUN preesistenti nel volume, Astra Trident imposterà il tipo di protezione predefinito su `CHAP` E procedere alla configurazione dell'iniziatore CHAP e del nome utente e dei segreti di destinazione.
  - Se la SVM contiene LUN, Astra Trident non attiverà CHAP sulla SVM. Ciò garantisce che l'accesso alle LUN già presenti sulla SVM non sia limitato.
- Configurazione dell'iniziatore CHAP e del nome utente e dei segreti di destinazione; queste opzioni devono essere specificate nella configurazione del backend (come mostrato sopra).
- Gestione dell'aggiunta di iniziatori a `igroupName` dato nel back-end. Se non specificato, l'impostazione predefinita è `trident`.

Una volta creato il backend, Astra Trident crea un corrispondente `tridentbackend` CRD e memorizza i segreti CHAP e i nomi utente come segreti Kubernetes. Tutti i PVS creati da Astra Trident su questo backend verranno montati e fissati su CHAP.

Ruota le credenziali e aggiorna i back-end

È possibile aggiornare le credenziali CHAP aggiornando i parametri CHAP in `backend.json` file. Per eseguire questa operazione, è necessario aggiornare i segreti CHAP e utilizzare `tridentctl update` per riflettere queste modifiche.

> Quando si aggiornano i segreti CHAP per un backend, è necessario utilizzare `tridentctl` per aggiornare il backend. Non aggiornare le credenziali sul cluster di storage attraverso l'interfaccia utente CLI/ONTAP, in quanto Astra Trident non sarà in grado di rilevare queste modifiche.

```
$ 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 |
+----------------+----------------+--------------------------------------
+--------+---------+
```

Le connessioni esistenti rimarranno inalterate; continueranno a rimanere attive se le credenziali vengono aggiornate da Astra Trident sulla SVM. Le nuove connessioni utilizzeranno le credenziali aggiornate e le connessioni esistenti continueranno a rimanere attive. Disconnettendo e riconnettendo il vecchio PVS, verranno utilizzate le credenziali aggiornate.

Opzioni di configurazione ed esempi

Scopri come creare e utilizzare i driver SAN ONTAP con l'installazione di Astra Trident. Questa sezione fornisce esempi di configurazione back-end e dettagli su come mappare i backend a StorageClasses.

Opzioni di configurazione back-end

Per le opzioni di configurazione del backend, consultare la tabella seguente:

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di storage	"ontap-nas", "ontap-nas-economy", "ontap-nas-flexgroup", "ontap-san", "ontap-san-economy"
`backendName`	Nome personalizzato o backend dello storage	Nome del driver + "_" + dataLIF
`managementLIF`	Indirizzo IP di un cluster o LIF di gestione SVM	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Indirizzo IP del protocollo LIF. USA le parentesi quadre per IPv6. Non può essere aggiornato dopo l'impostazione	Derivato dalla SVM, se non specificato
`useCHAP`	Utilizzare CHAP per autenticare iSCSI per i driver SAN ONTAP [booleano]	falso
`chapInitiatorSecret`	Segreto iniziatore CHAP. Necessario se `useCHAP=true`	""
`labels`	Set di etichette arbitrarie formattate con JSON da applicare sui volumi	""
`chapTargetInitiatorSecret`	CHAP target Initiator secret. Necessario se `useCHAP=true`	""
`chapUsername`	Nome utente inbound. Necessario se `useCHAP=true`	""
`chapTargetUsername`	Nome utente di destinazione. Necessario se `useCHAP=true`	""
`clientCertificate`	Valore del certificato client codificato con base64. Utilizzato per l'autenticazione basata su certificato	""
`clientPrivateKey`	Valore codificato in base64 della chiave privata del client. Utilizzato per l'autenticazione basata su certificato	""

Parametro	Descrizione	Predefinito
`trustedCACertificate`	Valore codificato in base64 del certificato CA attendibile. Opzionale. Utilizzato per l'autenticazione basata su certificato	""
`username`	Nome utente per la connessione al cluster/SVM. Utilizzato per l'autenticazione basata sulle credenziali	""
`password`	Password per la connessione al cluster/SVM. Utilizzato per l'autenticazione basata sulle credenziali	""
`svm`	Macchina virtuale per lo storage da utilizzare	Derivato se un SVM `managementLIF` è specificato
`igroupName`	Nome dell'igroup per i volumi SAN da utilizzare	"Trident-<backend-UUID>"
`storagePrefix`	Prefisso utilizzato per il provisioning di nuovi volumi nella SVM. Non può essere aggiornato dopo l'impostazione	"tridente"
`limitAggregateUsage`	Il provisioning non riesce se l'utilizzo è superiore a questa percentuale. **Non si applica ad Amazon FSX per ONTAP**	"" (non applicato per impostazione predefinita)
`limitVolumeSize`	Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore.	"" (non applicato per impostazione predefinita)
`lunsPerFlexvol`	LUN massimi per FlexVol, devono essere compresi nell'intervallo [50, 200]	"100"
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Ad esempio, {"api":false,} method":true	nullo

Per comunicare con il cluster ONTAP, è necessario fornire i parametri di autenticazione. Potrebbe trattarsi del nome utente/password di un account di accesso di sicurezza o di un certificato installato.

> Se si utilizza un backend Amazon FSX per NetApp ONTAP, non specificare `limitAggregateUsage` parametro. Il `fsxadmin` e. `vsadmin` I ruoli forniti da Amazon FSX per NetApp ONTAP non contengono le autorizzazioni di accesso necessarie per recuperare l'utilizzo aggregato e limitarlo tramite Astra Trident.

> Non utilizzare `debugTraceFlags` a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.

Per `ontap-san` Driver, l'impostazione predefinita prevede l'utilizzo di tutti gli IP LIF dei dati dalla SVM e l'utilizzo di multipath iSCSI. Specifica di un indirizzo IP per il dataLIF per `ontap-san` i driver li costringono a disattivare multipath e a utilizzare solo l'indirizzo specificato.

> Quando si crea un backend, ricordarlo `dataLIF` e. `storagePrefix` impossibile modificare dopo la creazione. Per aggiornare questi parametri, è necessario creare un nuovo backend.

`igroupName` Può essere impostato su un igroup già creato nel cluster ONTAP. Se non specificato, Astra Trident crea automaticamente un igroup denominato Trident-<backend-UUID>. Se si fornisce un igroupName predefinito, NetApp consiglia di utilizzare un igroup per cluster Kubernetes, se la SVM deve essere condivisa tra gli ambienti. Ciò è necessario affinché Astra Trident mantenga automaticamente aggiunte/eliminazioni IQN.

I back-end possono anche aggiornare igroups dopo la creazione:

- IgroupName può essere aggiornato per indicare un nuovo igroup creato e gestito sulla SVM all'esterno di Astra Trident.
- IgroupName può essere omesso. In questo caso, Astra Trident creerà e gestirà automaticamente un igroup trident-<backend-UUID>.

In entrambi i casi, gli allegati dei volumi continueranno ad essere accessibili. I futuri allegati dei volumi utilizzeranno l'igroup aggiornato. Questo aggiornamento non interrompe l'accesso ai volumi presenti nel back-end.

È possibile specificare un FQDN (Fully-qualified domain name) per `managementLIF` opzione.

```
`managementLIF` Per tutti i driver ONTAP è possibile impostare anche gli
indirizzi IPv6. Assicurarsi di installare Trident con `--use-ipv6`
allarme. È necessario prestare attenzione alla definizione `managementLIF`
Indirizzo IPv6 tra parentesi quadre.
```

> Quando si utilizzano indirizzi IPv6, assicurarsi `managementLIF` e. `dataLIF` (se incluso nella definizione del backend) sono definiti tra parentesi quadre, ad esempio [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Se `dataLIF` Non è fornito, Astra Trident recupererà i dati IPv6 LIF da SVM.

Per abilitare i driver ontap-san a utilizzare CHAP, impostare `useCHAP` parametro a. `true` nella definizione di back-end. Astra Trident configurerà e utilizzerà CHAP bidirezionale come autenticazione predefinita per la SVM fornita nel backend. Vedere "qui" per scoprire come funziona.

Per `ontap-san-economy` driver, il `limitVolumeSize` L'opzione limita inoltre le dimensioni massime dei volumi gestiti per qtree e LUN.

> Astra Trident imposta le etichette di provisioning nel campo "commenti" di tutti i volumi creati utilizzando `ontap-san` driver. Per ogni volume creato, il campo "commenti" di FlexVol contiene tutte le etichette presenti sul pool di storage in cui è inserito. Gli amministratori dello storage possono definire le etichette per ogni pool di storage e raggruppare tutti i volumi creati in un pool di storage. In questo modo è possibile differenziare i volumi in base a una serie di etichette personalizzabili fornite nella configurazione di back-end.

Opzioni di configurazione back-end per il provisioning dei volumi

Per impostazione predefinita, è possibile controllare il provisioning di ciascun volume utilizzando queste opzioni in una sezione speciale della configurazione. Per un esempio, vedere gli esempi di configurazione riportati di seguito.

Parametro	Descrizione	Predefinito
`spaceAllocation`	Allocazione dello spazio per LUN	"vero"
`spaceReserve`	Modalità di riserva dello spazio; "nessuno" (sottile) o "volume" (spesso)	"nessuno"
`snapshotPolicy`	Policy di Snapshot da utilizzare	"nessuno"
`qosPolicy`	Gruppo di criteri QoS da assegnare per i volumi creati. Scegliere tra qosPolicy o adaptiveQosPolicy per pool di storage/backend	""
`adaptiveQosPolicy`	Gruppo di criteri QoS adattivi da assegnare per i volumi creati. Scegliere tra qosPolicy o adaptiveQosPolicy per pool di storage/backend	""
`snapshotReserve`	Percentuale di volume riservato agli snapshot "0"	Se `snapshotPolicy` è "nessuno", altrimenti ""
`splitOnClone`	Separare un clone dal suo padre al momento della creazione	"falso"
`splitOnClone`	Separare un clone dal suo padre al momento della creazione	"falso"
`encryption`	Abilitare la crittografia dei volumi NetApp	"falso"
`securityStyle`	Stile di sicurezza per nuovi volumi	"unix"
`tieringPolicy`	Policy di tiering per utilizzare "nessuno"	"Solo snapshot" per configurazione SVM-DR precedente a ONTAP 9.5

> L'utilizzo di gruppi di policy QoS con Astra Trident richiede ONTAP 9.8 o versione successiva. Si consiglia di utilizzare un gruppo di criteri QoS non condiviso e assicurarsi che il gruppo di criteri sia applicato a ciascun componente singolarmente. Un gruppo di policy QoS condiviso applicherà il limite massimo per il throughput totale di tutti i carichi di lavoro.

Ecco un esempio con i valori predefiniti definiti:

```
{
 "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"
 }
}
```

> Per tutti i volumi creati utilizzando `ontap-san` Driver, Astra Trident aggiunge una capacità extra del 10% a FlexVol per ospitare i metadati LUN. Il LUN viene fornito con le dimensioni esatte richieste dall'utente nel PVC. Astra Trident aggiunge il 10% al FlexVol (viene visualizzato come dimensione disponibile in ONTAP). A questo punto, gli utenti otterranno la quantità di capacità utilizzabile richiesta. Questa modifica impedisce inoltre che le LUN diventino di sola lettura, a meno che lo spazio disponibile non sia completamente utilizzato. Ciò non si applica a ontap-san-Economy.

Per i backend che definiscono `snapshotReserve`, Astra Trident calcola le dimensioni dei volumi come segue:

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

Il 1.1 è il 10% aggiuntivo che Astra Trident aggiunge a FlexVol per ospitare i metadati LUN. Per `snapshotReserve` = 5% e richiesta PVC = 5GiB, la dimensione totale del volume è 5,79GiB e la dimensione disponibile è 5,5GiB. Il `volume show` il comando dovrebbe mostrare risultati simili a questo esempio:

```
Vserver   Volume       Aggregate    State      Type       Size  Available Used%
--------- ------------ ------------ ---------- ---- ---------- ---------- -----
                  _pvc_89f1c156_3801_4de4_9f9d_034d54c395f4
                                   online     RW         10GB     5.00GB    0%
                  _pvc_e42ec6fe_3baa_4af6_996d_134adbbb8e6d
                                   online     RW       5.79GB     5.50GB    0%
                  _pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba
                                   online     RW          1GB    511.8MB    0%
3 entries were displayed.
```

Attualmente, il ridimensionamento è l'unico modo per utilizzare il nuovo calcolo per un volume esistente.

Esempi di configurazione minimi

Gli esempi seguenti mostrano le configurazioni di base che lasciano la maggior parte dei parametri predefiniti. Questo è il modo più semplice per definire un backend.

> Se si utilizza Amazon FSX su NetApp ONTAP con Astra Trident, si consiglia di specificare i nomi DNS per i file LIF anziché gli indirizzi IP.

`ontap-san` driver con autenticazione basata su certificato

Si tratta di un esempio minimo di configurazione di back-end. `clientCertificate`, `clientPrivateKey`, e. `trustedCACertificate` (Facoltativo, se si utilizza una CA attendibile) sono inseriti in `backend.json` E prendere rispettivamente i valori codificati base64 del certificato client, della chiave privata e del certificato CA attendibile.

```
{
    "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` Driver con CHAP bidirezionale

Si tratta di un esempio minimo di configurazione di back-end. Questa configurazione di base crea un `ontap-san` back-end con `useCHAP` impostare su `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` **driver**

```
{
    "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"
}
```

Esempi di backend con pool di storage virtuali

Nel file di definizione back-end di esempio mostrato di seguito, vengono impostati valori predefiniti specifici per tutti i pool di storage, ad esempio `spaceReserve` a nessuno, `spaceAllocation` a false, e. `encryption` a falso. I pool di storage virtuali sono definiti nella sezione storage.

In questo esempio, alcuni dei pool di storage vengono impostati in modo personalizzato `spaceReserve`, `spaceAllocation`, e. `encryption` e alcuni pool sovrascrivono i valori predefiniti precedentemente impostati.

```
{
    "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"
            }
        }
    ]
}
```

Di seguito viene riportato un esempio iSCSI per `ontap-san-economy` driver:

```
{
    "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"
            }
        }
    ]
}
```

Mappare i backend in StorageClasses

Le seguenti definizioni di StorageClass si riferiscono ai pool di storage virtuali sopra indicati. Utilizzando il `parameters.selector` Ciascun StorageClass richiama i pool virtuali che possono essere utilizzati per ospitare un volume. Gli aspetti del volume saranno definiti nel pool virtuale scelto.

- Il primo StorageClass (`protection-gold`) verrà mappato al primo, secondo pool di storage virtuale in `ontap-nas-flexgroup` il back-end e il primo pool di storage virtuale in `ontap-san` back-end. Si tratta dell'unico pool che offre una protezione di livello gold.
- Il secondo StorageClass (`protection-not-gold`) verrà mappato al terzo e quarto pool di storage virtuale in `ontap-nas-flexgroup` back-end e il secondo, terzo pool di storage virtuale in `ontap-san` back-end. Questi sono gli unici pool che offrono un livello di protezione diverso dall'oro.
- Il terzo StorageClass (`app-mysqldb`) verrà mappato al quarto pool di storage virtuale in `ontap-nas` il back-end e il terzo pool di storage virtuale in `ontap-san-economy` back-end. Questi sono gli unici pool che offrono la configurazione del pool di storage per applicazioni di tipo mysqldb.
- Il quarto StorageClass (`protection-silver-creditpoints-20k`) verrà mappato al terzo pool di storage virtuale in `ontap-nas-flexgroup` il back-end e il secondo pool di storage virtuale in `ontap-san` back-end. Questi sono gli unici pool che offrono una protezione di livello gold a 20000 punti di credito.
- Quinta StorageClass (`creditpoints-5k`) verrà mappato al secondo pool di storage virtuale in `ontap-nas-economy` il back-end e il terzo pool di storage virtuale in `ontap-san` back-end. Queste sono le uniche offerte di pool a 5000 punti di credito.

Astra Trident deciderà quale pool di storage virtuale è selezionato e garantirà il rispetto dei requisiti di storage.

```
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"
```

Configurare un backend con i driver NAS ONTAP

Scopri come configurare un backend ONTAP con i driver NAS ONTAP.

- "Preparazione"
- "Configurazione ed esempi"

Autorizzazioni utente

Astra Trident prevede di essere eseguito come amministratore di ONTAP o SVM, in genere utilizzando `admin` utente del cluster o un `vsadmin` Utente SVM o un utente con un nome diverso che ha lo stesso ruolo. Per le implementazioni di Amazon FSX per NetApp ONTAP, Astra Trident prevede di essere eseguito come amministratore di ONTAP o SVM, utilizzando il cluster `fsxadmin` utente o a. `vsadmin` Utente SVM o un utente con un nome diverso che ha lo stesso ruolo. Il `fsxadmin` user è un sostituto limitato per l'utente amministratore del cluster.

> Se si utilizza `limitAggregateUsage` parametro, sono richieste le autorizzazioni di amministrazione del cluster. Quando si utilizza Amazon FSX per NetApp ONTAP con Astra Trident, il `limitAggregateUsage` il parametro non funziona con `vsadmin` e. `fsxadmin` account utente. L'operazione di configurazione non riesce se si specifica questo parametro.

Preparazione

Scopri come preparare la configurazione di un backend ONTAP con i driver NAS ONTAP. Per tutti i backend ONTAP, Astra Trident richiede almeno un aggregato assegnato alla SVM.

Per tutti i backend ONTAP, Astra Trident richiede almeno un aggregato assegnato alla SVM.

È inoltre possibile eseguire più di un driver e creare classi di storage che puntino all'una o all'altra. Ad esempio, è possibile configurare una classe Gold che utilizza `ontap-nas` Driver e una classe Bronze che utilizza `ontap-nas-economy` uno.

Tutti i nodi di lavoro di Kubernetes devono avere installati gli strumenti NFS appropriati. Vedere "qui" per ulteriori dettagli.

Autenticazione

Astra Trident offre due modalità di autenticazione di un backend ONTAP.

- Basato sulle credenziali: Nome utente e password di un utente ONTAP con le autorizzazioni richieste. Si consiglia di utilizzare un ruolo di accesso di sicurezza predefinito, ad esempio `admin` oppure `vsadmin` Per garantire la massima compatibilità con le versioni di ONTAP.
- Basato su certificato: Astra Trident può anche comunicare con un cluster ONTAP utilizzando un certificato installato sul backend. In questo caso, la definizione di backend deve contenere i valori codificati in Base64 del certificato client, della chiave e del certificato CA attendibile, se utilizzato (consigliato).

Gli utenti possono anche scegliere di aggiornare i back-end esistenti, scegliendo di passare da basato su credenziali a basato su certificato e viceversa. Se **vengono forniti sia credenziali che certificati**, Astra Trident utilizza per impostazione predefinita i certificati mentre emette un avviso per rimuovere le credenziali dalla definizione di backend.

Abilitare l'autenticazione basata su credenziali

Astra Trident richiede le credenziali di un amministratore con ambito SVM/cluster per comunicare con il backend ONTAP. Si consiglia di utilizzare ruoli standard predefiniti, ad esempio `admin` oppure `vsadmin`. Ciò garantisce la compatibilità con le future release di ONTAP che potrebbero esporre le API delle funzionalità da utilizzare nelle future release di Astra Trident. È possibile creare e utilizzare un ruolo di accesso di sicurezza personalizzato con Astra Trident, ma non è consigliato.

Una definizione di back-end di esempio avrà un aspetto simile al seguente:

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

Tenere presente che la definizione di backend è l'unica posizione in cui le credenziali vengono memorizzate in testo normale. Una volta creato il backend, i nomi utente e le password vengono codificati con Base64 e memorizzati come segreti Kubernetes. La creazione/l'updation di un backend è l'unico passaggio che richiede la conoscenza delle credenziali. Pertanto, si tratta di un'operazione di sola amministrazione, che deve essere eseguita dall'amministratore Kubernetes/storage.

Abilitare l'autenticazione basata su certificato

I backend nuovi ed esistenti possono utilizzare un certificato e comunicare con il backend ONTAP. Nella definizione di backend sono necessari tre parametri.

- ClientCertificate: Valore del certificato client codificato con base64.
- ClientPrivateKey: Valore codificato in base64 della chiave privata associata.
- TrustedCACertificate: Valore codificato in base64 del certificato CA attendibile. Se si utilizza una CA attendibile, è necessario fornire questo parametro. Questa operazione può essere ignorata se non viene utilizzata alcuna CA attendibile.

Un workflow tipico prevede i seguenti passaggi.

Fasi

1. Generare un certificato e una chiave del client. Durante la generazione, impostare il nome comune (CN) sull'utente ONTAP per l'autenticazione come.

   ```
   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. Aggiungere un certificato CA attendibile al cluster ONTAP. Questo potrebbe essere già gestito

dall'amministratore dello storage. Ignorare se non viene utilizzata alcuna CA attendibile.

```
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. Installare il certificato e la chiave del client (dal passaggio 1) sul cluster ONTAP.

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

4. Verificare che il ruolo di accesso di sicurezza di ONTAP supporti `cert` metodo di autenticazione.

```
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. Verifica dell'autenticazione utilizzando il certificato generato. Sostituire <LIF di gestione ONTAP> e <vserver name> con IP LIF di gestione e nome SVM. Assicurarsi che la politica di servizio di LIF sia impostata su `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. Codifica certificato, chiave e certificato CA attendibile con Base64.

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

7. Creare il backend utilizzando i valori ottenuti dal passaggio precedente.

```
$ 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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Aggiornare i metodi di autenticazione o ruotare le credenziali

È possibile aggiornare un backend esistente per utilizzare un metodo di autenticazione diverso o per ruotare le credenziali. Questo funziona in entrambi i modi: I backend che utilizzano il nome utente/la password possono essere aggiornati per utilizzare i certificati; i backend che utilizzano i certificati possono essere aggiornati in base al nome utente/alla password. A tale scopo, utilizzare un aggiornamento `backend.json` file contenente i parametri da eseguire `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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

> Quando si ruotano le password, l'amministratore dello storage deve prima aggiornare la password per l'utente su ONTAP. Seguito da un aggiornamento back-end. Durante la rotazione dei certificati, è possibile aggiungere più certificati all'utente. Il backend viene quindi aggiornato per utilizzare il nuovo certificato, dopodiché il vecchio certificato può essere cancellato dal cluster ONTAP.

L'aggiornamento di un backend non interrompe l'accesso ai volumi già creati, né influisce sulle connessioni dei volumi effettuate successivamente. Un aggiornamento back-end corretto indica che Astra Trident può comunicare con il backend ONTAP e gestire le future operazioni sui volumi.

Gestire le policy di esportazione NFS

Astra Trident utilizza policy di esportazione NFS per controllare l'accesso ai volumi forniti dall'IT.

Astra Trident offre due opzioni quando si lavora con le policy di esportazione:

- Astra Trident è in grado di gestire dinamicamente la policy di esportazione; in questa modalità operativa, l'amministratore dello storage specifica un elenco di blocchi CIDR che rappresentano indirizzi IP consentiti. Astra Trident aggiunge automaticamente gli IP dei nodi che rientrano in questi intervalli ai criteri di esportazione. In alternativa, se non viene specificato alcun CIDR, qualsiasi IP unicast con ambito globale trovato nei nodi verrà aggiunto alla policy di esportazione.

- Gli amministratori dello storage possono creare una policy di esportazione e aggiungere regole manualmente. Astra Trident utilizza il criterio di esportazione predefinito, a meno che nella configurazione non venga specificato un nome diverso del criterio di esportazione.

Gestione dinamica delle policy di esportazione

La versione 20.04 di CSI Trident offre la possibilità di gestire dinamicamente le policy di esportazione per i backend ONTAP. In questo modo, l'amministratore dello storage può specificare uno spazio di indirizzi consentito per gli IP dei nodi di lavoro, invece di definire manualmente regole esplicite. Semplifica notevolmente la gestione delle policy di esportazione; le modifiche alle policy di esportazione non richiedono più l'intervento manuale sul cluster di storage. Inoltre, questo consente di limitare l'accesso al cluster di storage solo ai nodi di lavoro che hanno IP nell'intervallo specificato, supportando una gestione dettagliata e automatica.

> La gestione dinamica delle policy di esportazione è disponibile solo per CSI Trident. È importante assicurarsi che i nodi di lavoro non vengano sottoposti a NATing.

Esempio

È necessario utilizzare due opzioni di configurazione. Ecco un esempio di definizione back-end:

```
{
    "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
}
```

> Quando si utilizza questa funzione, è necessario assicurarsi che la giunzione root di SVM disponga di un criterio di esportazione precreato con una regola di esportazione che consenta il blocco CIDR del nodo (ad esempio il criterio di esportazione predefinito). Seguire sempre le Best practice consigliate da NetApp per dedicare una SVM ad Astra Trident.

Ecco una spiegazione del funzionamento di questa funzione utilizzando l'esempio precedente:

- `autoExportPolicy` è impostato su `true`. Questo indica che Astra Trident creerà un criterio di esportazione per `svm1` SVM e gestire l'aggiunta e l'eliminazione di regole utilizzando `autoExportCIDRs` blocchi di indirizzi. Ad esempio, un backend con UUID 403b5326-8482-40db-96d0-d83fb3f4daec e. `autoExportPolicy` impostare su `true` crea un criterio di esportazione denominato `trident-403b5326-8482-40db-96d0-d83fb3f4daec` Su SVM.
- `autoExportCIDRs` contiene un elenco di blocchi di indirizzi. Questo campo è opzionale e per impostazione predefinita è ["0.0.0.0/0", "::/0"]. Se non definito, Astra Trident aggiunge tutti gli indirizzi unicast con ambito globale trovati nei nodi di lavoro.

In questo esempio, il `192.168.0.0/24` viene fornito uno spazio per gli indirizzi. Ciò indica che gli IP dei nodi Kubernetes che rientrano in questo intervallo di indirizzi verranno aggiunti alla policy di esportazione creata da Astra Trident. Quando Astra Trident registra un nodo su cui viene eseguito, recupera gli indirizzi IP del nodo e li confronta con i blocchi di indirizzo forniti in `autoExportCIDRs`. Dopo aver filtrato gli IP, Astra Trident crea regole di policy di esportazione per gli IP client individuati, con una regola per ogni nodo identificato.

È possibile eseguire l'aggiornamento `autoExportPolicy` e. `autoExportCIDRs` per i backend dopo la creazione. È possibile aggiungere nuovi CIDR a un backend gestito automaticamente o eliminare i CIDR esistenti. Prestare attenzione quando si eliminano i CIDR per assicurarsi che le connessioni esistenti non vengano interrotte. È anche possibile scegliere di disattivare `autoExportPolicy` per un backend e tornare a una policy di esportazione creata manualmente. Questa operazione richiede l'impostazione di `exportPolicy` nella configurazione del backend.

Dopo che Astra Trident ha creato o aggiornato un backend, è possibile controllare il backend utilizzando `tridentctl` o il corrispondente `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
```

Quando i nodi vengono aggiunti a un cluster Kubernetes e registrati con il controller Astra Trident, le policy di esportazione dei backend esistenti vengono aggiornate (a condizione che rientrino nell'intervallo di indirizzi specificato nella `autoExportCIDRs` per il back-end).

Quando un nodo viene rimosso, Astra Trident controlla tutti i backend in linea per rimuovere la regola di accesso per il nodo. Rimuovendo questo IP del nodo dalle policy di esportazione dei backend gestiti, Astra Trident impedisce i montaggi non autorizzati, a meno che questo IP non venga riutilizzato da un nuovo nodo nel cluster.

Per i backend esistenti in precedenza, aggiornare il backend con `tridentctl update backend` Garantisce che Astra Trident gestisca automaticamente le policy di esportazione. In questo modo si crea una nuova policy di esportazione denominata dopo l'UUID del backend e i volumi presenti sul backend utilizzeranno la policy di

esportazione appena creata una volta rimontati.

> L'eliminazione di un backend con policy di esportazione gestite automaticamente elimina la policy di esportazione creata dinamicamente. Se il backend viene ricreato, viene trattato come un nuovo backend e si otterrà la creazione di una nuova policy di esportazione.

Se l'indirizzo IP di un nodo live viene aggiornato, è necessario riavviare il pod Astra Trident sul nodo. Astra Trident aggiornerà quindi la policy di esportazione per i backend che riesce a riflettere questa modifica IP.

Opzioni di configurazione ed esempi

Scopri come creare e utilizzare i driver NAS ONTAP con l'installazione di Astra Trident. Questa sezione fornisce esempi di configurazione back-end e dettagli su come mappare i backend a StorageClasses.

Opzioni di configurazione back-end

Per le opzioni di configurazione del backend, consultare la tabella seguente:

Parametro	Descrizione	Predefinito
`version`		Sempre 1
`storageDriverName`	Nome del driver di storage	"ontap-nas", "ontap-nas-economy", "ontap-nas-flexgroup", "ontap-san", "ontap-san-economy"
`backendName`	Nome personalizzato o backend dello storage	Nome del driver + "_" + dataLIF
`managementLIF`	Indirizzo IP di un cluster o LIF di gestione SVM	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Indirizzo IP del protocollo LIF. USA le parentesi quadre per IPv6. Non può essere aggiornato dopo l'impostazione	Derivato dalla SVM, se non specificato
`autoExportPolicy`	Abilitare la creazione e l'aggiornamento automatici dei criteri di esportazione [booleano]	falso
`autoExportCIDRs`	Elenco di CIDR per filtrare gli IP dei nodi Kubernetes rispetto a quando `autoExportPolicy` è attivato	["0.0.0.0/0", ":/0"]`
`labels`	Set di etichette arbitrarie formattate con JSON da applicare sui volumi	""
`clientCertificate`	Valore del certificato client codificato con base64. Utilizzato per l'autenticazione basata su certificato	""
`clientPrivateKey`	Valore codificato in base64 della chiave privata del client. Utilizzato per l'autenticazione basata su certificato	""

Parametro	Descrizione	Predefinito
`trustedCACertificate`	Valore codificato in base64 del certificato CA attendibile. Opzionale. Utilizzato per l'autenticazione basata su certificato	""
`username`	Nome utente per la connessione al cluster/SVM. Utilizzato per l'autenticazione basata sulle credenziali	
`password`	Password per la connessione al cluster/SVM. Utilizzato per l'autenticazione basata sulle credenziali	
`svm`	Macchina virtuale per lo storage da utilizzare	Derivato se un SVM `managementLIF` è specificato
`igroupName`	Nome dell'igroup per i volumi SAN da utilizzare	"Trident-<backend-UUID>"
`storagePrefix`	Prefisso utilizzato per il provisioning di nuovi volumi nella SVM. Non può essere aggiornato dopo l'impostazione	"tridente"
`limitAggregateUsage`	Il provisioning non riesce se l'utilizzo è superiore a questa percentuale. **Non si applica ad Amazon FSX per ONTAP**	"" (non applicato per impostazione predefinita)
`limitVolumeSize`	Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore.	"" (non applicato per impostazione predefinita)
`lunsPerFlexvol`	LUN massimi per FlexVol, devono essere compresi nell'intervallo [50, 200]	"100"
`debugTraceFlags`	Flag di debug da utilizzare per la risoluzione dei problemi. Ad esempio, {"api":false,} method":true	nullo
`nfsMountOptions`	Elenco separato da virgole delle opzioni di montaggio NFS	""
`qtreesPerFlexvol`	Qtree massimi per FlexVol, devono essere compresi nell'intervallo [50, 300]	"200"
`useREST`	Parametro booleano per l'utilizzo delle API REST di ONTAP. **Anteprima tecnica**	falso

> `useREST` viene fornito come **anteprima tecnica** consigliata per ambienti di test e non per carichi di lavoro di produzione. Quando è impostato su `true`, Astra Trident utilizzerà le API REST di ONTAP per comunicare con il backend. Questa funzione richiede ONTAP 9.8 e versioni successive. Inoltre, il ruolo di accesso ONTAP utilizzato deve avere accesso a. `ontap` applicazione. Ciò è soddisfatto dal predefinito `vsadmin` e. `cluster-admin` ruoli.

Per comunicare con il cluster ONTAP, è necessario fornire i parametri di autenticazione. Potrebbe trattarsi del nome utente/password di un account di accesso di sicurezza o di un certificato installato.

> Se si utilizza un backend Amazon FSX per NetApp ONTAP, non specificare `limitAggregateUsage` parametro. Il `fsxadmin` e. `vsadmin` I ruoli forniti da Amazon FSX per NetApp ONTAP non contengono le autorizzazioni di accesso necessarie per recuperare l'utilizzo aggregato e limitarlo tramite Astra Trident.

> Non utilizzare `debugTraceFlags` a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.

> Quando si crea un backend, tenere presente che il `dataLIF` e. `storagePrefix` impossibile modificare dopo la creazione. Per aggiornare questi parametri, è necessario creare un nuovo backend.

È possibile specificare un FQDN (Fully-qualified domain name) per `managementLIF` opzione. È inoltre possibile specificare un FQDN per `dataLIF` In questo caso, l'FQDN verrà utilizzato per le operazioni di montaggio NFS. In questo modo è possibile creare un DNS round-robin per il bilanciamento del carico tra più LIF di dati.

```
`managementLIF` Per tutti i driver ONTAP è possibile impostare anche gli
indirizzi IPv6. Assicurarsi di installare Astra Trident con `--use-ipv6`
allarme. È necessario prestare attenzione alla definizione di
`managementLIF` Indirizzo IPv6 tra parentesi quadre.
```

> Quando si utilizzano indirizzi IPv6, assicurarsi `managementLIF` e. `dataLIF` (se incluso nella definizione del backend) sono definiti tra parentesi quadre, ad esempio [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Se `dataLIF` Non è fornito, Astra Trident recupererà i dati IPv6 LIF da SVM.

Utilizzando il `autoExportPolicy` e. `autoExportCIDRs` CSI Trident è in grado di gestire automaticamente le policy di esportazione. Questo è supportato per tutti i driver ontap-nas-*.

Per `ontap-nas-economy` driver, il `limitVolumeSize` L'opzione limita inoltre le dimensioni massime dei volumi gestiti per qtree e LUN e l' `qtreesPerFlexvol` Consente di personalizzare il numero massimo di qtree per FlexVol.

Il `nfsMountOptions` il parametro può essere utilizzato per specificare le opzioni di montaggio. Le opzioni di montaggio per i volumi persistenti di Kubernetes sono normalmente specificate nelle classi di storage, ma se non sono specificate opzioni di montaggio in una classe di storage, Astra Trident tornerà a utilizzare le opzioni di montaggio specificate nel file di configurazione del backend di storage. Se non sono specificate opzioni di montaggio nella classe di storage o nel file di configurazione, Astra Trident non imposta alcuna opzione di montaggio su un volume persistente associato.

Astra Trident imposta le etichette di provisioning nel campo "commenti" di tutti i volumi creati con(`ontap-nas` e.(`ontap-nas-flexgroup`. In base al driver utilizzato, i commenti vengono impostati su FlexVol (`ontap-nas`) O FlexGroup (`ontap-nas-flexgroup`). Astra Trident copia tutte le etichette presenti in un pool di storage nel volume di storage al momento del provisioning. Gli amministratori dello storage possono definire le etichette per ogni pool di storage e raggruppare tutti i volumi creati in un pool di storage. In questo modo è possibile differenziare i volumi in base a una serie di etichette personalizzabili fornite nella configurazione di back-end.

Opzioni di configurazione back-end per il provisioning dei volumi

Per impostazione predefinita, è possibile controllare il provisioning di ciascun volume utilizzando queste opzioni in una sezione speciale della configurazione. Per un esempio, vedere gli esempi di configurazione riportati di seguito.

Parametro	Descrizione	Predefinito
`spaceAllocation`	Allocazione dello spazio per LUN	"vero"
`spaceReserve`	Modalità di riserva dello spazio; "nessuno" (sottile) o "volume" (spesso)	"nessuno"
`snapshotPolicy`	Policy di Snapshot da utilizzare	"nessuno"
`qosPolicy`	Gruppo di criteri QoS da assegnare per i volumi creati. Scegliere tra qosPolicy o adaptiveQosPolicy per pool di storage/backend	""
`adaptiveQosPolicy`	Gruppo di criteri QoS adattivi da assegnare per i volumi creati. Scegliere tra qosPolicy o adaptiveQosPolicy per pool di storage/backend. Non supportato da ontap-nas-Economy.	""
`snapshotReserve`	Percentuale di volume riservato agli snapshot "0"	Se `snapshotPolicy` è "nessuno", altrimenti ""
`splitOnClone`	Separare un clone dal suo padre al momento della creazione	"falso"
`encryption`	Abilitare la crittografia dei volumi NetApp	"falso"
`securityStyle`	Stile di sicurezza per nuovi volumi	"unix"
`tieringPolicy`	Policy di tiering per utilizzare "nessuno"	"Solo snapshot" per configurazione SVM-DR precedente a ONTAP 9.5
UnixPermissions	Per i nuovi volumi	"777"
SnapshotDir	Controlla la visibilità di `.snapshot` directory	"falso"
ExportPolicy	Policy di esportazione da utilizzare	"predefinito"
SecurityStyle	Stile di sicurezza per nuovi volumi	"unix"

L'utilizzo di gruppi di policy QoS con Astra Trident richiede ONTAP 9.8 o versione successiva. Si consiglia di utilizzare un gruppo di criteri QoS non condiviso e assicurarsi che il gruppo di criteri sia applicato a ciascun componente singolarmente. Un gruppo di policy QoS condiviso applicherà il limite massimo per il throughput totale di tutti i carichi di lavoro.

Ecco un esempio con i valori predefiniti definiti:

```
{
  "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"
  }
}
```

Per `ontap-nas` e. `ontap-nas-flexgroups`, Astra Trident utilizza ora un nuovo calcolo per garantire che il FlexVol sia dimensionato correttamente con la percentuale di snapshotReserve e PVC. Quando l'utente richiede un PVC, Astra Trident crea il FlexVol originale con più spazio utilizzando il nuovo calcolo. Questo calcolo garantisce che l'utente riceva lo spazio scrivibile richiesto nel PVC e non uno spazio inferiore a quello richiesto. Prima della versione 21.07, quando l'utente richiede un PVC (ad esempio, 5GiB), con SnapshotReserve al 50%, ottiene solo 2,5 GiB di spazio scrivibile. Questo perché ciò che l'utente ha richiesto è l'intero volume e. `snapshotReserve` è una percentuale. Con Trident 21.07, ciò che l'utente richiede è lo spazio scrivibile e Astra Trident definisce `snapshotReserve` numero come percentuale dell'intero volume. Questo non si applica a. `ontap-nas-economy`. Vedere l'esempio seguente per vedere come funziona:

Il calcolo è il seguente:

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

Per snapshotReserve = 50% e richiesta PVC = 5GiB, la dimensione totale del volume è 2/0,5 = 10GiB e la

dimensione disponibile è 5GiB, che è ciò che l'utente ha richiesto nella richiesta PVC. Il `volume show` il comando dovrebbe mostrare risultati simili a questo esempio:

```
Vserver   Volume       Aggregate    State      Type       Size  Available Used%
--------- ------------ ------------ ---------- ---- ---------- ---------- -----
                 _pvc_89f1c156_3801_4de4_9f9d_034d54c395f4
                                      online     RW         10GB     5.00GB    0%
                 _pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba
                                      online     RW          1GB    511.8MB    0%
2 entries were displayed.
```

I backend esistenti delle installazioni precedenti eseguiranno il provisioning dei volumi come spiegato in precedenza durante l'aggiornamento di Astra Trident. Per i volumi creati prima dell'aggiornamento, è necessario ridimensionare i volumi per osservare la modifica. Ad esempio, un PVC 2GiB con `snapshotReserve=50` In precedenza, si è creato un volume che fornisce 1 GB di spazio scrivibile. Il ridimensionamento del volume su 3GiB, ad esempio, fornisce all'applicazione 3GiB di spazio scrivibile su un volume da 6 GiB.

Esempi di configurazione minimi

Gli esempi seguenti mostrano le configurazioni di base che lasciano la maggior parte dei parametri predefiniti. Questo è il modo più semplice per definire un backend.

> Se si utilizza Amazon FSX su NetApp ONTAP con Trident, si consiglia di specificare i nomi DNS per le LIF anziché gli indirizzi IP.

`ontap-nas` driver con autenticazione basata su certificato

Si tratta di un esempio minimo di configurazione di back-end. `clientCertificate`, `clientPrivateKey`, e. `trustedCACertificate` (Facoltativo, se si utilizza una CA attendibile) sono inseriti in `backend.json` E prendere rispettivamente i valori codificati base64 del certificato client, della chiave privata e del certificato CA attendibile.

```
{
  "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` driver con policy di esportazione automatica

Questo esempio mostra come impostare Astra Trident a utilizzare policy di esportazione dinamiche per creare e gestire automaticamente le policy di esportazione. Questo funziona allo stesso modo per `ontap-nas-`

`economy` e. `ontap-nas-flexgroup` driver.

```
{
    "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` **driver**

{ "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", "username": "vsadmin", "password": "secret", }

`ontap-nas` **Driver con 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` **driver**

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

Esempi di backend con pool di storage virtuali

Nel file di definizione back-end di esempio mostrato di seguito, vengono impostati valori predefiniti specifici per tutti i pool di storage, ad esempio `spaceReserve` a nessuno, `spaceAllocation` a false, e. `encryption` a falso. I pool di storage virtuali sono definiti nella sezione storage.

In questo esempio, alcuni dei pool di storage vengono impostati in modo personalizzato `spaceReserve`, `spaceAllocation`, e. `encryption` e alcuni pool sovrascrivono i valori predefiniti precedentemente impostati.

`ontap-nas` **driver**

```
{
    {
    "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` **driver**

```
{
    "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` **driver**

```
{
    "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"
            }
        }
    ]
}
```

Mappare i backend in StorageClasses

Le seguenti definizioni di StorageClass si riferiscono ai pool di storage virtuali sopra indicati. Utilizzando il `parameters.selector` Ciascun StorageClass richiama i pool virtuali che possono essere utilizzati per ospitare un volume. Gli aspetti del volume saranno definiti nel pool virtuale scelto.

- Il primo StorageClass (`protection-gold`) verrà mappato al primo, secondo pool di storage virtuale in `ontap-nas-flexgroup` il back-end e il primo pool di storage virtuale in `ontap-san` back-end. Si tratta dell'unico pool che offre una protezione di livello gold.
- Il secondo StorageClass (`protection-not-gold`) verrà mappato al terzo e quarto pool di storage virtuale in `ontap-nas-flexgroup` back-end e il secondo, terzo pool di storage virtuale in `ontap-san` back-end. Questi sono gli unici pool che offrono un livello di protezione diverso dall'oro.
- Il terzo StorageClass (`app-mysqldb`) verrà mappato al quarto pool di storage virtuale in `ontap-nas` il back-end e il terzo pool di storage virtuale in `ontap-san-economy` back-end. Questi sono gli unici pool che offrono la configurazione del pool di storage per applicazioni di tipo mysqldb.
- Il quarto StorageClass (`protection-silver-creditpoints-20k`) verrà mappato al terzo pool di storage virtuale in `ontap-nas-flexgroup` il back-end e il secondo pool di storage virtuale in `ontap-san` back-end. Questi sono gli unici pool che offrono una protezione di livello gold a 20000 punti di credito.
- Quinta StorageClass (`creditpoints-5k`) verrà mappato al secondo pool di storage virtuale in `ontap-nas-economy` il back-end e il terzo pool di storage virtuale in `ontap-san` back-end. Queste sono le uniche offerte di pool a 5000 punti di credito.

Astra Trident deciderà quale pool di storage virtuale è selezionato e garantirà il rispetto dei requisiti di storage.

```
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"
```

Utilizza Astra Trident con Amazon FSX per NetApp ONTAP

"Amazon FSX per NetApp ONTAP", È un servizio AWS completamente gestito che consente ai clienti di lanciare ed eseguire file system basati sul sistema operativo per lo storage ONTAP di NetApp. Amazon FSX per NetApp ONTAP ti consente di sfruttare le funzionalità, le performance e le funzionalità amministrative di NetApp che conosci, sfruttando al contempo la semplicità, l'agilità, la sicurezza e la scalabilità dell'archiviazione dei dati su AWS. FSX supporta molte delle funzionalità del file system e delle API di amministrazione di ONTAP.

Un file system è la risorsa principale di Amazon FSX, simile a un cluster ONTAP on-premise. All'interno di ogni SVM è possibile creare uno o più volumi, ovvero contenitori di dati che memorizzano i file e le cartelle nel file system. Con Amazon FSX per NetApp ONTAP, Data ONTAP verrà fornito come file system gestito nel cloud. Il nuovo tipo di file system è denominato **NetApp ONTAP**.

Utilizzando Astra Trident con Amazon FSX per NetApp ONTAP, è possibile garantire che i cluster Kubernetes in esecuzione in Amazon Elastic Kubernetes Service (EKS) siano in grado di eseguire il provisioning di volumi persistenti di blocchi e file supportati da ONTAP.

Scopri Astra Trident

Se sei un nuovo utente di Astra Trident, puoi familiarizzare con i link riportati di seguito:

- "FAQ"
- "Requisiti per l'utilizzo di Astra Trident"
- "Implementare Astra Trident"
- "Best practice per la configurazione di ONTAP, Cloud Volumes ONTAP e Amazon FSX per NetApp ONTAP"
- "Integrare Astra Trident"
- "Configurazione backend SAN ONTAP"
- "Configurazione backend NAS ONTAP"

Scopri di più sulle funzionalità dei driver "qui".

Amazon FSX per NetApp ONTAP utilizza "FabricPool" per gestire i tier di storage. Consente di memorizzare i dati in un Tier, in base all'accesso frequente ai dati.

Astra Trident prevede di essere eseguito come amministratore di ONTAP o SVM, utilizzando il cluster `fsxadmin` utente o a. `vsadmin` Utente SVM o un utente con un nome diverso che ha lo stesso ruolo. Il `fsxadmin` l'utente è un sostituto limitato di `admin` utente del cluster. Astra Trident utilizza in genere `admin` Utente del cluster per implementazioni non Amazon FSX per ONTAP.

Driver

Puoi integrare Astra Trident con Amazon FSX per NetApp ONTAP utilizzando i seguenti driver:

- `ontap-san`: Ogni PV fornito è un LUN all'interno del proprio volume Amazon FSX per NetApp ONTAP.
- `ontap-san-economy`: Ogni PV fornito è un LUN con un numero configurabile di LUN per volume Amazon FSX per NetApp ONTAP.
- `ontap-nas`: Ogni PV fornito è un volume Amazon FSX completo per NetApp ONTAP.
- `ontap-nas-economy`: Ogni PV fornito è un qtree, con un numero configurabile di qtree per ogni volume

Amazon FSX per NetApp ONTAP.

- `ontap-nas-flexgroup`: Ogni PV fornito è un volume Amazon FSX completo per NetApp ONTAP FlexGroup.

Autenticazione

Astra Trident offre due modalità di autenticazione:

- Basato sulle credenziali: È possibile utilizzare `fsxadmin` utente per il file system o l' `vsadmin` Configurato dall'utente per la SVM. Si consiglia di utilizzare `vsadmin` utente per configurare il backend. Astra Trident comunicherà con il file system FSX utilizzando questo nome utente e la password.
- Basato su certificato: Astra Trident comunicherà con SVM sul file system FSX utilizzando un certificato installato sulla SVM.

Per ulteriori informazioni sull'autenticazione, consulta i seguenti "NAS ONTAP"
* "ONTAP SAN"

Implementa e configura Astra Trident su EKS con Amazon FSX per NetApp ONTAP

Di cosa hai bisogno

- Un cluster Amazon EKS esistente o un cluster Kubernetes autogestito con `kubectl` installato.
- Una macchina virtuale per lo storage e il file system Amazon FSX per NetApp ONTAP, raggiungibile dai nodi di lavoro del cluster.
- Nodi di lavoro preparati per "NFS e/o iSCSI".

> Assicurati di seguire la procedura di preparazione del nodo richiesta per Amazon Linux e Ubuntu "Immagini Amazon Machine" (Amis) a seconda del tipo di AMI EKS.

Per altri requisiti di Astra Trident, vedere "qui".

Fasi

1. Implementare Astra Trident utilizzando uno dei ../trident-get-started/kubernetes-deploy.html[metodi di implementazione^].
2. Configurare Astra Trident come segue:
   a. Raccogliere il nome DNS LIF di gestione della SVM. Ad esempio, utilizzando l'interfaccia CLI AWS, individuare `DNSName` voce sotto `Endpoints` → `Management` dopo aver eseguito il seguente comando:

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

3. Creare e installare certificati per l'autenticazione. Se si utilizza un `ontap-san` back-end, vedere "qui". Se si utilizza un `ontap-nas` back-end, vedere "qui".

> È possibile accedere al file system (ad esempio per installare i certificati) utilizzando SSH da qualsiasi punto del file system. Utilizzare `fsxadmin` User (utente), la password configurata al momento della creazione del file system e il nome DNS di gestione da `aws fsx describe-file-systems`.

4. Creare un file backend utilizzando i certificati e il nome DNS della LIF di gestione, come mostrato nell'esempio seguente:

```
{
  "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",
 }
```

Per informazioni sulla creazione di backend, consulta i seguenti "Configurare un backend con i driver NAS ONTAP"
* "Configurare un backend con i driver SAN ONTAP"

> Non specificare `dataLIF` per `ontap-san` e. `ontap-san-economy` Driver per consentire ad Astra Trident di utilizzare multipath.

> Quando si utilizza Amazon FSX per NetApp ONTAP con Astra Trident, il `limitAggregateUsage` il parametro non funziona con `vsadmin` e. `fsxadmin` account utente. L'operazione di configurazione non riesce se si specifica questo parametro.

Dopo l'implementazione, eseguire la procedura per creare un "classe di storage, provisioning di un volume e montaggio del volume in un pod".

Trova ulteriori informazioni

- "Documentazione di Amazon FSX per NetApp ONTAP"
- "Post del blog su Amazon FSX per NetApp ONTAP"

Crea backend con kubectl

Un backend definisce la relazione tra Astra Trident e un sistema storage. Spiega ad Astra Trident come comunicare con quel sistema storage e come Astra Trident dovrebbe eseguire il provisioning dei volumi da esso. Dopo aver installato Astra Trident, il passo successivo è quello di creare un backend. Il `TridentBackendConfig` Custom Resource Definition (CRD) consente di creare e gestire backend Trident direttamente attraverso l'interfaccia Kubernetes. Per eseguire questa operazione, utilizzare `kubectl` O il tool CLI equivalente per la distribuzione Kubernetes.

TridentBackendConfig

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`) È un CRD front-end, namespace, che consente di gestire i backend Astra Trident utilizzando `kubectl`. Kubernetes e gli amministratori dello storage possono ora creare e gestire i backend direttamente attraverso la CLI di Kubernetes senza richiedere un'utility a riga di comando dedicata (`tridentctl`).

Alla creazione di un `TridentBackendConfig` oggetto, si verifica quanto segue:

- Astra Trident crea automaticamente un backend in base alla configurazione che fornisci. Questo è rappresentato internamente come a. `TridentBackend` (`tbe`, `tridentbackend`) CR.
- Il `TridentBackendConfig` è vincolato in modo univoco a un `TridentBackend` Creato da Astra Trident.

Ciascuno `TridentBackendConfig` mantiene una mappatura uno a uno con un `TridentBackend`. Il primo è l'interfaccia fornita all'utente per progettare e configurare i backend; il secondo è il modo in cui Trident rappresenta l'oggetto backend effettivo.

> `TridentBackend` I CRS vengono creati automaticamente da Astra Trident. Non è possibile modificarle. Se si desidera aggiornare i backend, modificare il `TridentBackendConfig` oggetto.

Vedere l'esempio seguente per il formato di `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
```

È inoltre possibile esaminare gli esempi in "trident-installer" directory per configurazioni di esempio per la piattaforma/servizio di storage desiderato.

Il `spec` utilizza parametri di configurazione specifici per il back-end. In questo esempio, il backend utilizza `ontap-san` storage driver e utilizza i parametri di configurazione riportati in tabella. Per l'elenco delle opzioni di configurazione per il driver di storage desiderato, consultare "informazioni di configurazione back-end per il driver di storage".

Il `spec` la sezione include anche `credentials` e. `deletionPolicy` i campi, che sono stati introdotti di recente in `TridentBackendConfig` CR:

- `credentials`: Questo parametro è un campo obbligatorio e contiene le credenziali utilizzate per l'autenticazione con il sistema/servizio di storage. Questo è impostato su un Kubernetes Secret creato dall'utente. Le credenziali non possono essere passate in testo normale e si verificherà un errore.
- `deletionPolicy`: Questo campo definisce cosa deve accadere quando `TridentBackendConfig` viene cancellato. Può assumere uno dei due valori possibili:
  - `delete`: Questo comporta l'eliminazione di entrambi `TridentBackendConfig` CR e il backend associato. Questo è il valore predefinito.
  - `retain`: Quando un `TridentBackendConfig` La CR viene eliminata, la definizione di back-end rimane presente e può essere gestita con `tridentctl`. Impostazione del criterio di eliminazione su `retain` consente agli utenti di eseguire il downgrade a una release precedente (precedente alla 21.04) e conservare i backend creati. Il valore di questo campo può essere aggiornato dopo un `TridentBackendConfig` viene creato.

> Il nome di un backend viene impostato utilizzando `spec.backendName`. Se non specificato, il nome del backend viene impostato sul nome di `TridentBackendConfig` oggetto (metadata.name). Si consiglia di impostare esplicitamente i nomi backend utilizzando `spec.backendName`.

> Backend creati con `tridentctl` non hanno un associato `TridentBackendConfig` oggetto. È possibile scegliere di gestire tali backend con `kubectl` creando un `TridentBackendConfig` CR. È necessario specificare parametri di configurazione identici (ad esempio `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`e così via). Astra Trident eseguirà automaticamente il binding del nuovo `TridentBackendConfig` con il backend preesistente.

Panoramica dei passaggi

Per creare un nuovo backend utilizzando `kubectl`, eseguire le seguenti operazioni:

1. Creare un "Kubernetes Secret". Il segreto contiene le credenziali che Astra Trident deve comunicare con il cluster/servizio di storage.
2. Creare un `TridentBackendConfig` oggetto. Contiene specifiche relative al cluster/servizio di storage e fa riferimento al segreto creato nel passaggio precedente.

Dopo aver creato un backend, è possibile osservarne lo stato utilizzando `kubectl get tbc <tbc-name> -n <trident-namespace>` e raccogliere ulteriori dettagli.

Fase 1: Creare un Kubernetes Secret

Creare un segreto contenente le credenziali di accesso per il backend. Si tratta di una caratteristica esclusiva di ogni piattaforma/servizio di storage. Ecco un esempio:

```
$ 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(>
```

Questa tabella riassume i campi che devono essere inclusi nel Secret per ciascuna piattaforma di storage:

Descrizione dei campi segreti della piattaforma di storage	Segreto	Descrizione dei campi
Azure NetApp Files	ID cliente	L'ID client dalla registrazione di un'applicazione
Cloud Volumes Service per AWS	ApiKey	Chiave API dell'account CVS
Cloud Volumes Service per AWS	SecretKey	Chiave segreta dell'account CVS
Cloud Volumes Service per GCP	id_chiave_privata	ID della chiave privata. Parte della chiave API per l'account di servizio GCP con ruolo di amministratore CVS
Cloud Volumes Service per GCP	private_key	Chiave privata. Parte della chiave API per l'account di servizio GCP con ruolo di amministratore CVS
Elemento (NetApp HCI/SolidFire)	Endpoint	MVIP per il cluster SolidFire con credenziali tenant
ONTAP	nome utente	Nome utente per la connessione al cluster/SVM. Utilizzato per l'autenticazione basata su credenziali
ONTAP	password	Password per la connessione al cluster/SVM. Utilizzato per l'autenticazione basata su credenziali

Descrizione dei campi segreti della piattaforma di storage	Segreto	Descrizione dei campi
ONTAP	ClientPrivateKey	Valore codificato in base64 della chiave privata del client. Utilizzato per l'autenticazione basata su certificato
ONTAP	ChapNomeUtente	Nome utente inbound. Obbligatorio se useCHAP=true. Per `ontap-san` e. `ontap-san-economy`
ONTAP	ChapInitialatorSecret	Segreto iniziatore CHAP. Obbligatorio se useCHAP=true. Per `ontap-san` e. `ontap-san-economy`
ONTAP	ChapTargetNomeUtente	Nome utente di destinazione. Obbligatorio se useCHAP=true. Per `ontap-san` e. `ontap-san-economy`
ONTAP	ChapTargetInitialatorSecret	CHAP target Initiator secret. Obbligatorio se useCHAP=true. Per `ontap-san` e. `ontap-san-economy`

Il Segreto creato in questo passaggio verrà indicato in `spec.credentials` campo di `TridentBackendConfig` oggetto creato nel passaggio successivo.

Fase 2: Creare `TridentBackendConfig` CR

A questo punto, è possibile creare il `TridentBackendConfig` CR. In questo esempio, un backend che utilizza `ontap-san` il driver viene creato utilizzando `TridentBackendConfig` oggetto mostrato di seguito:

```
$ 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
```

Fase 3: Verificare lo stato di `TridentBackendConfig` CR

Ora che è stato creato il `TridentBackendConfig` CR, è possibile verificare lo stato. Vedere il seguente esempio:

```
$ 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
```

Un backend è stato creato e associato a `TridentBackendConfig` CR.

La fase può assumere uno dei seguenti valori:

- `Bound`: Il `TridentBackendConfig` CR è associato a un backend e contiene tale backend `configRef` impostare su `TridentBackendConfig` L'uid di CR.
- `Unbound`: Rappresentato utilizzando `""`. Il `TridentBackendConfig` l'oggetto non è associato a un backend. Tutti creati di recente `TridentBackendConfig` I CRS sono in questa fase per impostazione predefinita. Una volta modificata la fase, non sarà più possibile tornare a Unbound.
- `Deleting`: Il `TridentBackendConfig` CR `deletionPolicy` è stato impostato per l'eliminazione. Quando il `TridentBackendConfig` La CR viene eliminata, passa allo stato di eliminazione.
  - Se non sono presenti richieste di rimborso di volumi persistenti (PVC) sul back-end, eliminare il `TridentBackendConfig` In questo modo Astra Trident elimina il backend e il `TridentBackendConfig` CR.
  - Se uno o più PVC sono presenti sul backend, passa a uno stato di eliminazione. Il `TridentBackendConfig` Successivamente, la CR entra anche nella fase di eliminazione. Il backend e. `TridentBackendConfig` Vengono eliminati solo dopo l'eliminazione di tutti i PVC.
- `Lost`: Il backend associato a `TridentBackendConfig` La CR è stata eliminata accidentalmente o deliberatamente e il `TridentBackendConfig` CR ha ancora un riferimento al backend cancellato. Il

`TridentBackendConfig` La CR può comunque essere eliminata indipendentemente da `deletionPolicy` valore.

- `Unknown`: Astra Trident non è in grado di determinare lo stato o l'esistenza del backend associato a `TridentBackendConfig` CR. Ad esempio, se il server API non risponde o se `tridentbackends.trident.netapp.io` CRD mancante. Ciò potrebbe richiedere l'intervento dell'utente.

In questa fase, viene creato un backend. È possibile gestire anche diverse operazioni, ad esempio "aggiornamenti back-end ed eliminazioni back-end".

(Facoltativo) fase 4: Ulteriori informazioni

È possibile eseguire il seguente comando per ottenere ulteriori informazioni sul backend:

```
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
```

Inoltre, è possibile ottenere un dump YAML/JSON di `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` contiene `backendName` e a. `backendUUID` del back-end creato in risposta a `TridentBackendConfig` CR. Il `lastOperationStatus` il campo rappresenta lo stato dell'ultima operazione di `TridentBackendConfig` CR, che può essere attivato dall'utente (ad esempio, l'utente ha modificato qualcosa in `spec`) O attivato da Astra Trident (ad esempio, durante il riavvio di Astra Trident). Può essere Success (riuscito) o Failed (non riuscito). `phase` rappresenta lo stato della relazione tra `TridentBackendConfig` CR e il back-end. Nell'esempio precedente, `phase` Ha il valore associato, il che significa che il `TridentBackendConfig` CR è associato al backend.

È possibile eseguire `kubectl -n trident describe tbc <tbc-cr-name>` per ottenere i dettagli dei registri degli eventi.

> Non è possibile aggiornare o eliminare un backend che contiene un associato `TridentBackendConfig` utilizzo di oggetti `tridentctl`. Comprendere le fasi necessarie per passare da un'operazione all'altra `tridentctl` e. `TridentBackendConfig`, "vedi qui".

Eseguire la gestione del back-end con kubectl

Scopri come eseguire operazioni di gestione back-end utilizzando `kubectl`.

Eliminare un backend

Eliminando un `TridentBackendConfig`, Si richiede ad Astra Trident di eliminare/conservare i backend (in base a. `deletionPolicy`). Per eliminare un backend, assicurarsi che `deletionPolicy` è impostato per eliminare. Per eliminare solo il `TridentBackendConfig`, assicurarsi che `deletionPolicy` è impostato su retain. In questo modo si garantisce che il backend sia ancora presente e che possa essere gestito tramite `tridentctl`.

Eseguire il seguente comando:

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

Astra Trident non elimina i Kubernetes Secrets utilizzati da `TridentBackendConfig`. L'utente Kubernetes è responsabile della pulizia dei segreti. Prestare attenzione quando si eliminano i segreti. È necessario eliminare i segreti solo se non vengono utilizzati dai backend.

Visualizzare i backend esistenti

Eseguire il seguente comando:

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

Puoi anche correre `tridentctl get backend -n trident` oppure `tridentctl get backend -o yaml -n trident` per ottenere un elenco di tutti i backend esistenti. Questo elenco includerà anche i backend creati con `tridentctl`.

Aggiornare un backend

Possono esserci diversi motivi per aggiornare un backend:

- Le credenziali del sistema storage sono state modificate. Per aggiornare le credenziali, il Kubernetes Secret utilizzato in `TridentBackendConfig` l'oggetto deve essere aggiornato. Astra Trident aggiornerà automaticamente il backend con le credenziali più recenti fornite. Eseguire il seguente comando per aggiornare Kubernetes Secret:

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

- È necessario aggiornare i parametri (ad esempio il nome della SVM ONTAP utilizzata). In questo caso, `TridentBackendConfig` Gli oggetti possono essere aggiornati direttamente tramite Kubernetes.

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

In alternativa, apportare modifiche all'esistente `TridentBackendConfig` CR eseguendo il seguente comando:

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

Se un aggiornamento back-end non riesce, il back-end continua a rimanere nella sua ultima configurazione nota. È possibile visualizzare i log per determinare la causa eseguendo `kubectl get tbc <tbc-name> -o yaml -n trident` oppure `kubectl describe tbc <tbc-name> -n trident`.

Dopo aver identificato e corretto il problema con il file di configurazione, è possibile eseguire nuovamente il comando update.

Eseguire la gestione back-end con tridentctl

Scopri come eseguire operazioni di gestione back-end utilizzando `tridentctl`.

Creare un backend

Dopo aver creato un "file di configurazione back-end", eseguire il seguente comando:

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

Se la creazione del back-end non riesce, si è verificato un errore nella configurazione del back-end. È possibile visualizzare i log per determinare la causa eseguendo il seguente comando:

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

Dopo aver identificato e corretto il problema con il file di configurazione, è possibile eseguire semplicemente `create` di nuovo comando.

Eliminare un backend

Per eliminare un backend da Astra Trident, procedere come segue:

1. Recuperare il nome del backend:

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

2. Eliminare il backend:

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

> Se Astra Trident ha eseguito il provisioning di volumi e snapshot da questo backend ancora esistenti, l'eliminazione del backend impedisce il provisioning di nuovi volumi da parte dell'IT. Il backend continuerà a esistere in uno stato di eliminazione e Trident continuerà a gestire tali volumi e snapshot fino a quando non verranno eliminati.

Visualizzare i backend esistenti

Per visualizzare i backend di cui Trident è a conoscenza, procedere come segue:

- Per ottenere un riepilogo, eseguire il seguente comando:

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

- Per ottenere tutti i dettagli, eseguire il seguente comando:

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

Aggiornare un backend

Dopo aver creato un nuovo file di configurazione back-end, eseguire il seguente comando:

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

Se l'aggiornamento del back-end non riesce, si è verificato un errore nella configurazione del back-end o si è tentato di eseguire un aggiornamento non valido. È possibile visualizzare i log per determinare la causa eseguendo il seguente comando:

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

Dopo aver identificato e corretto il problema con il file di configurazione, è possibile eseguire semplicemente `update` di nuovo comando.

Identificare le classi di storage che utilizzano un backend

Questo è un esempio del tipo di domande a cui puoi rispondere con il JSON che `tridentctl` output per oggetti backend. Viene utilizzato il `jq` che è necessario installare.

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

Questo vale anche per i backend creati con `TridentBackendConfig`.

Passare da un'opzione di gestione back-end all'altra

Scopri i diversi modi di gestire i backend in Astra Trident. Con l'introduzione di `TridentBackendConfig`, gli amministratori dispongono ora di due metodi unici per gestire i back-end. Questo pone le seguenti domande:

- È possibile creare backend utilizzando `tridentctl` essere gestito con `TridentBackendConfig`?
- È possibile creare backend utilizzando `TridentBackendConfig` essere gestito con `tridentctl`?

Gestire `tridentctl` backend con `TridentBackendConfig`

In questa sezione vengono descritte le procedure necessarie per gestire i backend creati con `tridentctl` Direttamente attraverso l'interfaccia Kubernetes creando `TridentBackendConfig` oggetti.

Questo si applica ai seguenti scenari:

- Backend preesistenti, che non dispongono di `TridentBackendConfig` perché sono stati creati con `tridentctl`.
- Nuovi backend creati con `tridentctl`, mentre altri `TridentBackendConfig` esistono oggetti.

In entrambi gli scenari, i backend continueranno a essere presenti, con Astra Trident che pianifica i volumi e li gestisce. Gli amministratori possono scegliere tra due opzioni:

- Continuare a utilizzare `tridentctl` per gestire i back-end creati utilizzando l'it.
- Collegare i backend creati con `tridentctl` a un nuovo `TridentBackendConfig` oggetto. In questo modo, i backend verranno gestiti utilizzando `kubectl` e non `tridentctl`.

Per gestire un backend preesistente utilizzando `kubectl`, sarà necessario creare un `TridentBackendConfig` che si collega al back-end esistente. Ecco una panoramica sul funzionamento di questo sistema:

1. Crea un Kubernetes Secret. Il segreto contiene le credenziali che Astra Trident deve comunicare con il cluster/servizio di storage.
2. Creare un `TridentBackendConfig` oggetto. Contiene specifiche relative al cluster/servizio di storage e fa riferimento al segreto creato nel passaggio precedente. È necessario specificare parametri di configurazione identici (ad esempio `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`e così via). `spec.backendName` deve essere impostato sul nome del backend esistente.

Fase 0: Identificare il backend

Per creare un `TridentBackendConfig` che si collega a un backend esistente, sarà necessario ottenere la configurazione del backend. In questo esempio, supponiamo che sia stato creato un backend utilizzando la seguente definizione JSON:

```
$ 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"
            }
        }
    ]
}
```

Fase 1: Creare un Kubernetes Secret

Creare un Segreto contenente le credenziali per il backend, come illustrato in questo esempio:

```
$ 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
```

Fase 2: Creare un `TridentBackendConfig` CR

Il passaggio successivo consiste nella creazione di un `TridentBackendConfig` CR che si associerà automaticamente al preesistente `ontap-nas-backend` (come in questo esempio). Assicurarsi che siano soddisfatti i seguenti requisiti:

- Lo stesso nome backend viene definito in `spec.backendName`.
- I parametri di configurazione sono identici al backend originale.
- I pool di storage virtuali (se presenti) devono mantenere lo stesso ordine del backend originale.
- Le credenziali vengono fornite attraverso un Kubernetes Secret e non in testo normale.

In questo caso, il `TridentBackendConfig` avrà un aspetto simile al seguente:

```
$ 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
```

Fase 3: Verificare lo stato di `TridentBackendConfig` CR

Dopo il `TridentBackendConfig` è stato creato, la sua fase deve essere `Bound`. Deve inoltre riflettere lo stesso nome e UUID del backend esistente.

```
$ 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 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

Il back-end verrà ora completamente gestito utilizzando `tbc-ontap-nas-backend` `TridentBackendConfig` oggetto.

Gestire `TridentBackendConfig` backend con `tridentctl`

```
`tridentctl` può essere utilizzato per elencare i backend creati con
`TridentBackendConfig`. Inoltre, gli amministratori possono anche
scegliere di gestire completamente tali backend attraverso `tridentctl`
eliminando `TridentBackendConfig` e assicurandosi `spec.deletionPolicy` è
impostato su `retain`.
```

Fase 0: Identificare il backend

Ad esempio, supponiamo che il seguente backend sia stato creato utilizzando `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 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

Dall'output, si vede che `TridentBackendConfig` È stato creato correttamente ed è associato a un backend [osservare l'UUID del backend].

Fase 1: Confermare `deletionPolicy` è impostato su `retain`

Diamo un'occhiata al valore di `deletionPolicy`. Questo valore deve essere impostato su `retain`. In questo modo si garantisce che quando si verifica un `TridentBackendConfig` La CR viene eliminata, la definizione di back-end rimane presente e può essere gestita con `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
```

Non passare alla fase successiva a meno che `deletionPolicy` è impostato su `retain`.

Fase 2: Eliminare `TridentBackendConfig` CR

Il passaggio finale consiste nell'eliminare `TridentBackendConfig` CR. Dopo la conferma di `deletionPolicy` è impostato su `retain`, è possibile procedere con l'eliminazione:

```
$ 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 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

Al momento dell'eliminazione di `TridentBackendConfig` Astra Trident lo rimuove senza eliminare il backend stesso.

Gestire le classi di storage

Informazioni sulla creazione di una classe di storage, sull'eliminazione di una classe di storage e sulla visualizzazione delle classi di storage esistenti.

Progettare una classe di storage

Vedere "qui" per ulteriori informazioni sulle classi di storage e su come configurarle.

Creare una classe di storage

Dopo aver creato un file di classe storage, eseguire il seguente comando:

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

`<storage-class-file>` deve essere sostituito con il nome del file della classe di storage.

Eliminare una classe di storage

Per eliminare una classe di storage da Kubernetes, eseguire il seguente comando:

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

`<storage-class>` deve essere sostituito con la classe di storage.

Tutti i volumi persistenti creati attraverso questa classe di storage resteranno inalterati e Astra Trident continuerà a gestirli.

> Astra Trident impone un vuoto `fsType` per i volumi creati. Per i backend iSCSI, si consiglia di applicare `parameters.fsType` In StorageClass. È necessario eliminare e ricreare StorageClasses con `parameters.fsType` specificato.

Visualizzare le classi di storage esistenti

- Per visualizzare le classi di storage Kubernetes esistenti, eseguire il seguente comando:

```
kubectl get storageclass
```

- Per visualizzare i dettagli della classe storage Kubernetes, eseguire il seguente comando:

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

- Per visualizzare le classi di storage sincronizzate di Astra Trident, eseguire il seguente comando:

```
tridentctl get storageclass
```

- Per visualizzare i dettagli della classe di storage sincronizzata di Astra Trident, eseguire il seguente comando:

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

Impostare una classe di storage predefinita

Kubernetes 1.6 ha aggiunto la possibilità di impostare una classe di storage predefinita. Si tratta della classe di storage che verrà utilizzata per eseguire il provisioning di un volume persistente se un utente non ne specifica uno in un PVC (Persistent Volume Claim).

- Definire una classe di storage predefinita impostando l'annotazione `storageclass.kubernetes.io/is-default-class` a true nella definizione della classe di storage. In base alla specifica, qualsiasi altro valore o assenza di annotazione viene interpretato come falso.
- È possibile configurare una classe di storage esistente come classe di storage predefinita utilizzando il seguente comando:

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

- Allo stesso modo, è possibile rimuovere l'annotazione predefinita della classe di storage utilizzando il seguente comando:

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

Nel bundle del programma di installazione di Trident sono presenti anche alcuni esempi che includono questa annotazione.

> Nel cluster dovrebbe essere presente una sola classe di storage predefinita per volta. Kubernetes non impedisce tecnicamente di averne più di una, ma si comporta come se non ci fosse alcuna classe di storage predefinita.

Identificare il backend per una classe di storage

Questo è un esempio del tipo di domande a cui puoi rispondere con il JSON che `tridentctl` Output per gli oggetti backend Astra Trident. Viene utilizzato il `jq` che potrebbe essere necessario installare per prima.

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

Eseguire operazioni sui volumi

Scopri le funzionalità offerte da Astra Trident per la gestione dei volumi.

- "Utilizzare la topologia CSI"
- "Lavorare con le istantanee"
- "Espandere i volumi"
- "Importa volumi"

Utilizzare la topologia CSI

Astra Trident può creare e collegare in modo selettivo volumi ai nodi presenti in un cluster Kubernetes utilizzando "Funzionalità topologia CSI". Utilizzando la funzionalità topologia CSI, l'accesso ai volumi può essere limitato a un sottoinsieme di nodi, in base alle aree geografiche e alle zone di disponibilità. I provider di cloud oggi consentono agli amministratori di Kubernetes di generare nodi basati su zone. I nodi possono trovarsi in diverse regioni all'interno di una zona di disponibilità o in varie zone di disponibilità. Per facilitare il provisioning dei volumi per i carichi di lavoro in un'architettura multi-zona, Astra Trident utilizza la topologia CSI.

> Scopri di più sulla funzionalità topologia CSI "qui".

Kubernetes offre due esclusive modalità di binding del volume:

- Con `VolumeBindingMode` impostare su `Immediate`, Astra Trident crea il volume senza alcuna consapevolezza della topologia. Il binding dei volumi e il provisioning dinamico vengono gestiti quando viene creato il PVC. Questa è l'impostazione predefinita `VolumeBindingMode` ed è adatto per i cluster che non applicano vincoli di topologia. I volumi persistenti vengono creati senza alcuna dipendenza dai requisiti di pianificazione del pod richiedente.
- Con `VolumeBindingMode` impostare su `WaitForFirstConsumer`, La creazione e il binding di un volume persistente per un PVC viene ritardata fino a quando un pod che utilizza il PVC viene pianificato e creato. In questo modo, i volumi vengono creati per soddisfare i vincoli di pianificazione imposti dai requisiti di topologia.

> Il `WaitForFirstConsumer` la modalità di binding non richiede etichette di topologia. Questo può essere utilizzato indipendentemente dalla funzionalità topologia CSI.

Di cosa hai bisogno

Per utilizzare la topologia CSI, è necessario disporre di quanto segue:

- Un cluster Kubernetes con 1.17 o versione successiva.

```
$ 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"}
```

- I nodi nel cluster devono essere dotati di etichette che introducano la consapevolezza della topologia (`topology.kubernetes.io/region` e. `topology.kubernetes.io/zone`). Queste etichette **devono essere presenti sui nodi del cluster** prima dell'installazione di Astra Trident affinché Astra Trident sia consapevole della topologia.

```
$ 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"}]
```

Fase 1: Creazione di un backend compatibile con la topologia

I backend di storage Astra Trident possono essere progettati per eseguire il provisioning selettivo dei volumi in base alle zone di disponibilità. Ogni backend può portare un optional `supportedTopologies` blocco che rappresenta un elenco di zone e regioni che devono essere supportate. Per StorageClasses che utilizzano tale backend, un volume viene creato solo se richiesto da un'applicazione pianificata in una regione/zona supportata.

Di seguito viene riportato un esempio di definizione di backend:

```
{
 "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` viene utilizzato per fornire un elenco di regioni e zone per backend. Queste regioni e zone rappresentano l'elenco dei valori consentiti che possono essere forniti in una StorageClass. Per StorageClasses che contengono un sottoinsieme delle regioni e delle zone fornite in un backend, Astra Trident creerà un volume sul backend.

È possibile definire `supportedTopologies` anche per pool di storage. Vedere il seguente esempio:

```
{"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 questo esempio, il `region` e. `zone` le etichette indicano la posizione del pool di storage. `topology.kubernetes.io/region` e. `topology.kubernetes.io/zone` stabilire da dove possono essere consumati i pool di storage.

Fase 2: Definire StorageClasses che siano compatibili con la topologia

In base alle etichette della topologia fornite ai nodi del cluster, è possibile definire StorageClasses in modo da contenere informazioni sulla topologia. In questo modo verranno determinati i pool di storage che fungono da candidati per le richieste PVC effettuate e il sottoinsieme di nodi che possono utilizzare i volumi forniti da Trident.

Vedere il seguente esempio:

```
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"
```

Nella definizione di StorageClass sopra riportata, `volumeBindingMode` è impostato su `WaitForFirstConsumer`. I PVC richiesti con questa classe di storage non verranno utilizzati fino a quando non saranno referenziati in un pod. Inoltre, `allowedTopologies` fornisce le zone e la regione da utilizzare. Il `netapp-san-us-east1` StorageClass crea PVC su `san-backend-us-east1` backend definito sopra.

Fase 3: Creare e utilizzare un PVC

Con StorageClass creato e mappato a un backend, è ora possibile creare PVC.

Vedere l'esempio `spec` sotto:

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

La creazione di un PVC utilizzando questo manifesto comporta quanto segue:

```
$ 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
```

Affinché Trident crei un volume e lo leghi al PVC, utilizza il PVC in un pod. Vedere il seguente esempio:

```
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
```

Questo podSpec indica a Kubernetes di pianificare il pod sui nodi presenti in `us-east1` e scegliere tra i nodi presenti in `us-east1-a` oppure `us-east1-b` zone.

Vedere il seguente output:

```
$ 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
```

Aggiorna i back-end da includere `supportedTopologies`

I backend preesistenti possono essere aggiornati per includere un elenco di `supportedTopologies` utilizzo di `tridentctl backend update`. Ciò non influisce sui volumi già sottoposti a provisioning e verrà utilizzato solo per i PVC successivi.

Trova ulteriori informazioni

- "Gestire le risorse per i container"
- "NodeSelector"
- "Affinità e anti-affinità"
- "Contamini e pedaggi"

Lavorare con le istantanee

A partire dalla versione 20.01 di Astra Trident, è possibile creare snapshot di PVS nel livello Kubernetes. È possibile utilizzare queste snapshot per mantenere copie point-in-time dei volumi creati da Astra Trident e pianificare la creazione di volumi aggiuntivi (cloni). Lo snapshot del volume è supportato da `ontap-nas`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `aws-cvs`, `gcp-cvs`, e. `azure-netapp-files` driver.

> Questa funzione è disponibile da Kubernetes 1.17 (beta) ed è GA da 1.20. Per informazioni sulle modifiche apportate al passaggio dalla versione beta a quella GA, vedere "il blog di release". Con la laurea in GA, il `v1` La versione API è stata introdotta ed è compatibile con le versioni precedenti `v1beta1` snapshot.

Di cosa hai bisogno

- La creazione di snapshot dei volumi richiede la creazione di un controller di snapshot esterno e di alcune definizioni di risorse personalizzate (CRD). Questa è la responsabilità del Kubernetes orchestrator in uso (ad esempio: Kubeadm, GKE, OpenShift).

È possibile creare uno snapshot-controller esterno e uno snapshot CRD come segue:

1. Creare CRD snapshot di volume:

```
$ 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. Creare lo snapshot-controller nello spazio dei nomi desiderato. Modificare i manifesti YAML riportati di seguito per modificare lo spazio dei nomi.

```
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 offre un "convalida di webhook" per aiutare gli utenti a convalidare le istantanee v1beta1 esistenti e confermare che si tratta di oggetti di risorse validi. Il webhook validante etichetta automaticamente gli oggetti snapshot non validi e impedisce la creazione di oggetti non validi futuri. Il webhook di convalida viene implementato da Kubernetes orchestrator. Consultare le istruzioni per implementare manualmente il webhook di convalida "qui". Trova esempi di manifesti di snapshot non validi "qui".

Nell'esempio riportato di seguito vengono illustrati i costrutti necessari per l'utilizzo delle snapshot e viene illustrato come creare e utilizzare le istantanee.

Fase 1: Impostare un `VolumeSnapshotClass`

Prima di creare un'istantanea del volume, impostare un collegamento:../trident-reference/objects.html[`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
```

Il `driver` Indica il driver CSI di Astra Trident. `deletionPolicy` può essere `Delete` oppure `Retain`. Quando è impostato su `Retain`, lo snapshot fisico sottostante sul cluster di storage viene conservato anche quando `VolumeSnapshot` oggetto eliminato.

Fase 2: Creare un'istantanea di un PVC esistente

```
$ 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
```

Lo snapshot è in fase di creazione per un PVC denominato `pvc1`e il nome dello snapshot è impostato su `pvc1-snap`.

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

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

Questo ha creato un `VolumeSnapshot` oggetto. Un'istantanea VolumeSnapshot è analoga a un PVC ed è associata a un `VolumeSnapshotContent` oggetto che rappresenta lo snapshot effettivo.

È possibile identificare `VolumeSnapshotContent` oggetto per `pvc1-snap` VolumeSnapshot descrivendolo.

```
$ 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
.
.
```

Il `Snapshot Content Name` Identifica l'oggetto VolumeSnapshotContent che fornisce questa snapshot. Il `Ready To Use` Il parametro indica che l'istantanea può essere utilizzata per creare un nuovo PVC.

Fase 3: Creazione di PVC da VolumeSnapshots

Vedere l'esempio seguente per creare un PVC utilizzando uno snapshot:

```
$ 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` Mostra che il PVC deve essere creato utilizzando un VolumeSnapshot denominato `pvc1-snap`

come origine dei dati. Questo indica ad Astra Trident di creare un PVC dall'istantanea. Una volta creato, il PVC può essere collegato a un pod e utilizzato come qualsiasi altro PVC.

> Quando si elimina un volume persistente con snapshot associate, il volume Trident corrispondente viene aggiornato a uno stato di eliminazione. Per eliminare il volume Astra Trident, è necessario rimuovere le snapshot del volume.

Trova ulteriori informazioni

- "Snapshot dei volumi"
- `VolumeSnapshotClass`

Espandere i volumi

Astra Trident offre agli utenti Kubernetes la possibilità di espandere i propri volumi dopo la loro creazione. Informazioni sulle configurazioni richieste per espandere i volumi iSCSI e NFS.

Espandere un volume iSCSI

È possibile espandere un volume persistente iSCSI (PV) utilizzando il provisioning CSI.

> L'espansione del volume iSCSI è supportata da `ontap-san`, `ontap-san-economy`, `solidfire-san` Driver e richiede Kubernetes 1.16 e versioni successive.

Panoramica

L'espansione di un PV iSCSI include i seguenti passaggi:

- Modifica della definizione di StorageClass per impostare `allowVolumeExpansion` campo a. `true`.
- Modifica della definizione PVC e aggiornamento di `spec.resources.requests.storage` per riflettere le nuove dimensioni desiderate, che devono essere superiori alle dimensioni originali.
- Il collegamento del PV deve essere collegato a un pod per poter essere ridimensionato. Esistono due scenari quando si ridimensiona un PV iSCSI:
  - Se il PV è collegato a un pod, Astra Trident espande il volume sul backend dello storage, esegue di nuovo la scansione del dispositivo e ridimensiona il file system.
  - Quando si tenta di ridimensionare un PV non collegato, Astra Trident espande il volume sul backend dello storage. Dopo aver associato il PVC a un pod, Trident esegue nuovamente la scansione del dispositivo e ridimensiona il file system. Kubernetes aggiorna quindi le dimensioni del PVC dopo il completamento dell'operazione di espansione.

L'esempio seguente mostra come funziona l'espansione del PVS iSCSI.

Fase 1: Configurare StorageClass per supportare l'espansione dei volumi

```
$ 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
```

Per un StorageClass già esistente, modificarlo per includere `allowVolumeExpansion` parametro.

Fase 2: Creare un PVC con la StorageClass creata

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

Astra Trident crea un volume persistente (PV) e lo associa a questo PVC (Persistent Volume Claim).

```
$ 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
```

Fase 3: Definire un pod che colleghi il PVC

In questo esempio, viene creato un pod che utilizza `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
```

Fase 4: Espandere il PV

Per ridimensionare il PV creato da 1 Gi a 2 Gi, modificare la definizione PVC e aggiornare `spec.resources.requests.storage` A 2 Gi.

```
$ 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
 ...
```

Fase 5: Convalidare l’espansione

È possibile verificare che l’espansione funzioni correttamente controllando le dimensioni del volume PVC, PV e Astra Trident:

```
$ 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    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

Espandere un volume NFS

Astra Trident supporta l'espansione dei volumi per NFS PVS su cui è stato eseguito il provisioning `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `aws-cvs`, `gcp-cvs`, e. `azure-netapp-files` back-end.

Fase 1: Configurare StorageClass per supportare l'espansione dei volumi

Per ridimensionare un PV NFS, l'amministratore deve prima configurare la classe di storage per consentire l'espansione del volume impostando `allowVolumeExpansion` campo a. `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
```

Se è già stata creata una classe di storage senza questa opzione, è possibile modificare semplicemente la classe di storage esistente utilizzando `kubectl edit storageclass` per consentire l'espansione del volume.

Fase 2: Creare un PVC con la StorageClass creata

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

Astra Trident deve creare un PV NFS 20MiB per questo PVC:

```
$ 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
```

Fase 3: Espandere il PV

Per ridimensionare il PV 20MiB appena creato in 1GiB, modificare il PVC e impostare `spec.resources.requests.storage` A 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
...
```

Fase 4: Convalidare l’espansione

È possibile verificare che il ridimensionamento funzioni correttamente controllando le dimensioni del volume PVC, PV e Astra Trident:

```
$ 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    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

Importa volumi

È possibile importare volumi di storage esistenti come PV Kubernetes utilizzando `tridentctl import`.

Driver che supportano l'importazione di volumi

Questa tabella illustra i driver che supportano l'importazione di volumi e la release in cui sono stati introdotti.

Driver	Rilasciare
`ontap-nas`	19.04
`ontap-nas-flexgroup`	19.04
`solidfire-san`	19.04
`aws-cvs`	19.04

Driver	Rilasciare
azure-netapp-files	19.04
gcp-cvs	19.04
ontap-san	19.04

Perché è necessario importare i volumi?

Esistono diversi casi di utilizzo per l'importazione di un volume in Trident:

- La creazione di un'applicazione e il riutilizzo del set di dati esistente
- Utilizzo di un clone di un set di dati per un'applicazione temporanea
- Ricostruzione di un cluster Kubernetes guasto
- Migrazione dei dati delle applicazioni durante il disaster recovery

Come funziona l'importazione?

Il file PVC (Persistent Volume Claim) viene utilizzato dal processo di importazione del volume per creare il PVC. Come minimo, il file PVC deve includere i campi name, namespace, accessModes e storageClassName, come illustrato nell'esempio seguente.

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

Il `tridentctl` client viene utilizzato per importare un volume di storage esistente. Trident importa il volume persistendo i metadati del volume e creando PVC e PV.

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

Per importare un volume di storage, specificare il nome del backend Astra Trident contenente il volume, nonché il nome che identifica in modo univoco il volume nello storage (ad esempio: ONTAP FlexVol, Volume elemento, percorso del volume CVS). Il volume di storage deve consentire l'accesso in lettura/scrittura ed essere accessibile dal backend Astra Trident specificato. Il `-f` L'argomento string è obbligatorio e specifica il percorso del file YAML o JSON PVC.

Quando Astra Trident riceve la richiesta del volume di importazione, le dimensioni del volume esistente

vengono determinate e impostate nel PVC. Una volta importato il volume dal driver di storage, il PV viene creato con un ClaimRef sul PVC. La policy di recupero viene inizialmente impostata su `retain` Nel PV. Dopo che Kubernetes ha eseguito il binding con PVC e PV, la policy di recupero viene aggiornata in modo da corrispondere alla policy di recupero della classe di storage. Se il criterio di recupero della classe di storage è `delete`, Il volume di storage viene cancellato quando il PV viene cancellato.

Quando viene importato un volume con `--no-manage` Argomento: Trident non esegue operazioni aggiuntive sul PVC o sul PV per il ciclo di vita degli oggetti. Perché Trident ignora gli eventi PV e PVC per `--no-manage` Oggetti, il volume di storage non viene cancellato quando il PV viene cancellato. Vengono ignorate anche altre operazioni, come il clone del volume e il ridimensionamento del volume. Questa opzione è utile se si desidera utilizzare Kubernetes per carichi di lavoro containerizzati, ma altrimenti si desidera gestire il ciclo di vita del volume di storage al di fuori di Kubernetes.

Al PVC e al PV viene aggiunta un'annotazione che serve a doppio scopo per indicare che il volume è stato importato e se il PVC e il PV sono gestiti. Questa annotazione non deve essere modificata o rimossa.

Trident 19.07 e versioni successive gestiscono il collegamento di PVS e montano il volume come parte dell'importazione. Per le importazioni che utilizzano versioni precedenti di Astra Trident, non verranno eseguite operazioni nel percorso dei dati e l'importazione del volume non verificherà se il volume può essere montato. Se si commette un errore con l'importazione del volume (ad esempio, StorageClass non è corretta), è possibile ripristinare modificando la policy di recupero sul PV in `retain`, Eliminando PVC e PV e riprovando il comando di importazione del volume.

`ontap-nas` e. `ontap-nas-flexgroup` importazioni

Ogni volume creato con `ontap-nas` Driver è un FlexVol sul cluster ONTAP. Importazione di FlexVol con `ontap-nas` il driver funziona allo stesso modo. Un FlexVol già presente in un cluster ONTAP può essere importato come `ontap-nas` PVC. Allo stesso modo, è possibile importare i volumi FlexGroup come `ontap-nas-flexgroup` PVC.

> Un volume ONTAP deve essere di tipo rw per essere importato da Trident. Se un volume è di tipo dp, si tratta di un volume di destinazione SnapMirror; è necessario interrompere la relazione di mirroring prima di importare il volume in Trident.

> Il `ontap-nas` il driver non può importare e gestire qtree. Il `ontap-nas` e. `ontap-nas-flexgroup` i driver non consentono nomi di volumi duplicati.

Ad esempio, per importare un volume denominato `managed_volume` su un backend denominato `ontap_nas`, utilizzare il seguente comando:

```
$ 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    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

Per importare un volume denominato `unmanaged_volume` (su `ontap_nas backend`), che Trident non gestirà, utilizzare il seguente comando:

```
$ 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   |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

Quando si utilizza `--no-manage` Argomento: Trident non rinomina il volume né convalida se il volume è stato montato. L'operazione di importazione del volume non riesce se il volume non è stato montato manualmente.

> È stato risolto un bug esistente in precedenza relativo all'importazione di volumi con UnixPermissions personalizzati. È possibile specificare unixPermissions nella definizione PVC o nella configurazione backend e richiedere ad Astra Trident di importare il volume di conseguenza.

`ontap-san` importa

Astra Trident può anche importare SAN FlexVol ONTAP che contengono una singola LUN. Ciò è coerente con `ontap-san` Driver, che crea un FlexVol per ogni PVC e un LUN all'interno di FlexVol. È possibile utilizzare `tridentctl import` comando nello stesso modo degli altri casi:

- Includere il nome di `ontap-san` back-end.

- Specificare il nome del FlexVol da importare. Tenere presente che questo FlexVol contiene un solo LUN che deve essere importato.
- Fornire il percorso della definizione PVC che deve essere utilizzata con `-f` allarme.
- Scegli tra gestire il PVC o non gestirlo. Per impostazione predefinita, Trident gestirà il PVC e rinominerà il FlexVol e il LUN sul backend. Per importare come volume non gestito, passare a. `--no-manage` allarme.

> Quando si importa un non gestito `ontap-san` Assicurarsi che il LUN nel FlexVol sia denominato `lun0` ed è mappato ad un igroup con gli iniziatori desiderati. Astra Trident gestisce automaticamente questa operazione per un'importazione gestita.

Astra Trident importa il FlexVol e lo associa alla definizione del PVC. Astra Trident rinomina anche FlexVol in `pvc-<uuid>` E il LUN all'interno di FlexVol a. `lun0`.

> Si consiglia di importare volumi che non dispongono di connessioni attive. Se si desidera importare un volume utilizzato attivamente, clonare prima il volume, quindi eseguire l'importazione.

Esempio

Per importare `ontap-san-managed` FlexVol presente su `ontap_san_default` eseguire il backend `tridentctl import` comando come:

```
$ 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    |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
```

> Per poter essere importato da Astra Trident, un volume ONTAP deve essere di tipo rw. Se un volume è di tipo dp, si tratta di un volume di destinazione SnapMirror; è necessario interrompere la relazione di mirroring prima di importare il volume in Astra Trident.

`element` importa

Con Trident è possibile importare il software NetApp Element/volumi NetApp HCI nel cluster Kubernetes. È necessario il nome del backend Astra Trident e il nome univoco del volume e del file PVC come argomenti per `tridentctl import` comando.

```
$ 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    |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
```

Il driver Element supporta nomi di volumi duplicati. Se sono presenti nomi di volumi duplicati, il processo di importazione dei volumi di Trident restituisce un errore. Come soluzione alternativa, clonare il volume e fornire un nome di volume univoco. Quindi importare il volume clonato.

`aws-cvs` **importa**

Per importare un volume supportato da NetApp Cloud Volumes Service in AWS, identificare il volume in base al percorso del volume anziché al nome.

Per importare un `aws-cvs` volume sul backend chiamato `awscvs_YEppr` con il percorso del volume di `adroit-jolly-swift`, utilizzare il seguente comando:

```
$ 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    |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
```

Il percorso del volume è la parte del percorso di esportazione del volume dopo :/. Ad esempio, se il percorso di esportazione è `10.0.0.1:/adroit-jolly-swift`, il percorso del volume è `adroit-jolly-swift`.

`gcp-cvs` importa

Importazione di un `gcp-cvs` il volume funziona come l'importazione di un `aws-cvs` volume.

`azure-netapp-files` importa

Per importare un `azure-netapp-files` volume sul backend chiamato `azurenetappfiles_40517` con il percorso del volume `importvol1`, eseguire il seguente comando:

```
$ 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    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

> Il percorso del volume per il volume ANF è presente nel percorso di montaggio dopo :/. Ad esempio, se il percorso di montaggio è `10.0.0.2:/importvol1`, il percorso del volume è `importvol1`.

Preparare il nodo di lavoro

Tutti i nodi di lavoro nel cluster Kubernetes devono essere in grado di montare i volumi di cui si è eseguito il provisioning per i pod. Se si utilizza `ontap-nas`, `ontap-nas-economy`, o. `ontap-nas-flexgroup` Driver per uno dei tuoi back-end, i nodi di lavoro hanno bisogno degli strumenti NFS. In caso contrario, richiedono gli strumenti iSCSI.

Le versioni recenti di RedHat CoreOS hanno sia NFS che iSCSI installati per impostazione predefinita.

> È necessario riavviare sempre i nodi di lavoro dopo l'installazione degli strumenti NFS o iSCSI, altrimenti il collegamento dei volumi ai container potrebbe non riuscire.

Volumi NFS

Protocollo	Sistema operativo	Comandi
NFS	RHEL/CentOS	`sudo yum install -y nfs-utils`

Protocollo	Sistema operativo	Comandi
NFS	Ubuntu/Debian	`sudo apt-get install -y nfs-common`

Assicurarsi che il servizio NFS venga avviato durante l'avvio.

Volumi iSCSI

Quando si utilizzano volumi iSCSI, considerare quanto segue:

- Ogni nodo del cluster Kubernetes deve avere un IQN univoco. **Questo è un prerequisito necessario**.
- Se si utilizza RHCOS versione 4.5 o successiva oppure RHEL o CentOS versione 8.2 o successiva con `solidfire-san` Verificare che l'algoritmo di autenticazione CHAP sia impostato su MD5 in `/etc/iscsi/iscsid.conf`.

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

- Quando si utilizzano nodi di lavoro che eseguono RHEL/RedHat CoreOS con iSCSI PVS, assicurarsi di specificare `discard` MountOption in StorageClass per eseguire la rigenerazione dello spazio inline. Vedere "La documentazione di RedHat".

Protocollo	Sistema operativo	Comandi

ISCSI	RHEL/CentOS	1. Installare i seguenti pacchetti di sistema:
		
		`sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-mapper-multipath`
		
		2. Verificare che la versione di iscsi-initiator-utils sia 6.2.0.874-2.el7 o successiva:
		
		`rpm -q iscsi-initiator-utils`
		
		3. Impostare la scansione su manuale:
		
		`sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/' /etc/iscsi/iscsid.conf`
		
		4. Abilitare il multipathing:
		
		`sudo mpathconf --enable --with_multipathd y`
		
		5. Assicurarsi che `iscsid` e. `multipathd` sono in esecuzione:
		
		`sudo systemctl enable --now iscsid multipathd`
		
		6. Attivare e avviare `iscsi`:
		
		`sudo systemctl enable --now iscsi`

Protocollo	Sistema operativo	Comandi

ISCSI	Ubuntu/Debian	1. Installare i seguenti pacchetti di sistema:
		
		`sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools scsitools`
		
		2. Verificare che la versione Open-iscsi sia 2.0.874-5ubuntu2.10 o successiva (per il bionico) o 2.0.874-7.1ubuntu6.1 o successiva (per il focale):
		
		`dpkg -l open-iscsi`
		
		3. Impostare la scansione su manuale:
		
		`sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/' /etc/iscsi/iscsid.conf`
		
		4. Abilitare il multipathing:
		
		`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. Assicurarsi che `open-iscsi` e. `multipath-tools` sono abilitati e in esecuzione:
		
		`sudo systemctl status multipath-tools`
		`sudo systemctl enable --now open-iscsi.service`
		`sudo systemctl status open-iscsi`

Per Ubuntu 18.04, è necessario rilevare le porte di destinazione con `iscsiadm` prima di iniziare `open-iscsi` Per avviare il daemon iSCSI. In alternativa, è possibile modificare `iscsi` servizio da avviare `iscsid` automaticamente.

Per ulteriori informazioni sulla preparazione automatica del nodo di lavoro, che è una funzionalità beta, vedere "qui".

Preparazione automatica del nodo di lavoro

Astra Trident può installare automaticamente il necessario `NFS` e. `iSCSI` Strumenti sui nodi presenti nel cluster Kubernetes. Si tratta di una funzionalità * beta* e non è prevista per* cluster di produzione. Attualmente, la funzionalità è disponibile per i nodi che eseguono **CentOS, RHEL e Ubuntu**.

Per questa funzionalità, Astra Trident include un nuovo flag di installazione: `--enable-node-prep` per le installazioni implementate con `tridentctl`. Per le implementazioni con l'operatore Trident, utilizzare l'opzione booleano `enableNodePrep`.

Il `--enable-node-prep` L'opzione di installazione indica ad Astra Trident di installare e assicurarsi che i pacchetti e/o i servizi NFS e iSCSI siano in esecuzione quando un volume viene montato su un nodo di lavoro. Si tratta di una funzionalità * beta* destinata all'utilizzo in ambienti di sviluppo/test **non qualificati** per l'utilizzo in produzione.

Quando il `--enable-node-prep` Flag incluso per le installazioni Astra Trident implementate con `tridentctl`, ecco cosa succede:

1. Nell'ambito dell'installazione, Astra Trident registra i nodi su cui viene eseguito.
2. Quando viene effettuata una richiesta di rimborso del volume persistente (PVC), Astra Trident crea un PV da uno dei backend gestiti.
3. L'utilizzo del PVC in un pod richiede ad Astra Trident di montare il volume sul nodo su cui viene eseguito il pod. Astra Trident tenta di installare le utility client NFS/iSCSI necessarie e di garantire che i servizi richiesti siano attivi. Questa operazione viene eseguita prima del montaggio del volume.

La preparazione di un nodo di lavoro viene eseguita una sola volta come parte del primo tentativo di montaggio di un volume. Tutti i successivi montaggi di volume dovrebbero avere successo, purché non vengano apportate modifiche all'esterno di Astra Trident `NFS` e. `iSCSI` utilità.

In questo modo, Astra Trident può garantire che tutti i nodi di un cluster Kubernetes dispongano delle utility necessarie per montare e collegare i volumi. Per i volumi NFS, la policy di esportazione dovrebbe anche consentire il montaggio del volume. Trident può gestire automaticamente le policy di esportazione per backend; in alternativa, gli utenti possono gestire le policy di esportazione fuori banda.

Monitorare Astra Trident

Astra Trident fornisce un set di endpoint di metriche Prometheus che è possibile utilizzare per monitorare le performance di Astra Trident.

Le metriche fornite da Astra Trident ti consentono di:

- Tieni sotto controllo lo stato di salute e la configurazione di Astra Trident. È possibile esaminare il successo delle operazioni e se è in grado di comunicare con i back-end come previsto.

- Esaminare le informazioni sull'utilizzo del back-end e comprendere il numero di volumi sottoposti a provisioning su un back-end, la quantità di spazio consumato e così via.
- Mantenere una mappatura della quantità di volumi forniti sui backend disponibili.
- Tenere traccia delle performance. Puoi dare un'occhiata a quanto tempo ci vuole per Astra Trident per comunicare con i back-end ed eseguire le operazioni.

> Per impostazione predefinita, le metriche di Trident sono esposte sulla porta di destinazione `8001` su `/metrics` endpoint. Queste metriche sono **abilitate per impostazione predefinita** quando Trident è installato.

Di cosa hai bisogno

- Un cluster Kubernetes con Astra Trident installato.
- Un'istanza Prometheus. Questo può essere un "Implementazione di Prometheus in container" Oppure puoi scegliere di eseguire Prometheus come a. "applicazione nativa".

Fase 1: Definire un target Prometheus

Devi definire un target Prometheus per raccogliere le metriche e ottenere informazioni sui backend gestiti da Astra Trident, sui volumi creati e così via. Questo "blog" Spiega come utilizzare Prometheus e Grafana con Astra Trident per recuperare le metriche. Il blog spiega come eseguire Prometheus come operatore nel cluster Kubernetes e come creare un ServiceMonitor per ottenere le metriche di Astra Trident.

Fase 2: Creazione di un ServiceMonitor Prometheus

Per utilizzare le metriche Trident, è necessario creare un ServiceMonitor Prometheus che controlli `trident-csi` e ascolta su `metrics` porta. Un esempio di ServiceMonitor è simile al seguente:

```
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
```

Questa definizione di ServiceMonitor recupera le metriche restituite da `trident-csi` e in particolare cerca di

`metrics` endpoint del servizio. Di conseguenza, Prometheus è ora configurato per comprendere le metriche di Astra Trident.

Oltre alle metriche disponibili direttamente da Astra Trident, Kubelet ne espone molte `kubelet_volume_*` metriche tramite il proprio endpoint di metriche. Kubelet può fornire informazioni sui volumi collegati, sui pod e sulle altre operazioni interne gestite. Vedere "qui".

Fase 3: Eseguire una query sulle metriche di Trident con PromQL

PromQL è utile per la creazione di espressioni che restituiscono dati di serie temporali o tabulari.

Di seguito sono riportate alcune query PromQL che è possibile utilizzare:

Ottieni informazioni sulla salute di Trident

- **Percentuale di risposte HTTP 2XX da Astra Trident**

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

- **Percentuale di risposte REST da Astra Trident tramite codice di stato**

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

- **Durata media in ms delle operazioni eseguite da Astra Trident**

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

Ottieni informazioni sull'utilizzo di Astra Trident

- **Dimensione media del volume**

```
trident_volume_allocated_bytes/trident_volume_count
```

- **Spazio totale del volume fornito da ciascun backend**

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

Ottieni l'utilizzo di singoli volumi

> Questa opzione è attivata solo se vengono raccolte anche le metriche del kubelet.

- **Percentuale di spazio utilizzato per ciascun volume**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Scopri di più sulla telemetria Astra Trident AutoSupport

Per impostazione predefinita, Astra Trident invia le metriche Prometheus e le informazioni di back-end di base a NetApp ogni giorno.

- Per impedire ad Astra Trident di inviare a NetApp le metriche Prometheus e le informazioni di back-end di base, passare il `--silence-autosupport` Segnalazione durante l'installazione di Astra Trident.
- Astra Trident può anche inviare i log dei container al NetApp Support on-demand tramite `tridentctl send autosupport`. Devi attivare Astra Trident per caricare i registri. Prima di inviare i log, è necessario accettare i file NetApphttps://www.netapp.com/company/legal/privacy-policy/["direttiva sulla privacy"^].
- Se non specificato, Astra Trident recupera i registri delle ultime 24 ore.
- È possibile specificare il periodo di conservazione dei log con `--since` allarme. Ad esempio: `tridentctl send autosupport --since=1h`. Queste informazioni vengono raccolte e inviate tramite un `trident-autosupport` Container installato insieme ad Astra Trident. È possibile ottenere l'immagine del contenitore in "Trident AutoSupport".
- Trident AutoSupport non raccoglie né trasmette dati personali o di identificazione personale (PII). Viene fornito con un "EULA" Non applicabile all'immagine del contenitore Trident. Scopri di più sull'impegno di NetApp per la sicurezza e la fiducia dei dati "qui".

Un payload di esempio inviato da Astra Trident è simile al seguente:

```
{
  "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
    }
  ]
}
```


- I messaggi AutoSupport vengono inviati all'endpoint AutoSupport di NetApp. Se si utilizza un registro privato per memorizzare le immagini container, è possibile utilizzare `--image-registry` allarme.
- È inoltre possibile configurare gli URL proxy generando i file YAML di installazione. Per eseguire questa operazione, utilizzare `tridentctl install --generate-custom-yaml` Per creare i file YAML e aggiungere `--proxy-url` argomento per `trident-autosupport` container in `trident-deployment.yaml`.

Disattiva le metriche di Astra Trident

Per **disattivare** il report delle metriche, è necessario generare YAML personalizzati (utilizzando il `--generate-custom-yaml` e modificarli per rimuovere `--metrics` il contrassegno di non essere richiamato per `trident-main`container.