



# Utilizzare Astra Trident

## Astra Trident

NetApp  
April 03, 2024

# Sommario

- Utilizzare Astra Trident ..... 1
- Preparare il nodo di lavoro ..... 1
- Configurare e gestire i backend ..... 6
- Creare e gestire classi di archiviazione ..... 126
- Provisioning e gestione dei volumi ..... 131

# Utilizzare Astra Trident

## Preparare il nodo di lavoro

Tutti i nodi di lavoro nel cluster Kubernetes devono essere in grado di montare i volumi forniti per i pod. Per preparare i nodi di lavoro, è necessario installare gli strumenti NFS, iSCSI o NVMe/TCP in base alla selezione del driver.

### Selezionare gli strumenti giusti

Se si utilizza una combinazione di driver, è necessario installare tutti gli strumenti necessari per i driver. Le versioni recenti di RedHat CoreOS hanno gli strumenti installati di default.

#### Strumenti NFS

Installa gli strumenti NFS se utilizzi: `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `azure-netapp-files`, `gcp-cvs`.

#### Strumenti iSCSI

Installare gli strumenti iSCSI se si utilizzano: `ontap-san`, `ontap-san-economy`, `solidfire-san`.

#### Strumenti NVMe

Installa gli strumenti NVMe se stai utilizzando `ontap-san` Per il protocollo NVMe (nonvolatile Memory Express) su TCP (NVMe/TCP).



Consigliamo ONTAP 9,12 o versione successiva per NVMe/TCP.

## Rilevamento del servizio del nodo

Astra Trident tenta di rilevare automaticamente se il nodo può eseguire servizi iSCSI o NFS.



Il rilevamento del servizio nodo identifica i servizi rilevati ma non garantisce che i servizi siano configurati correttamente. Al contrario, l'assenza di un servizio rilevato non garantisce il mancato funzionamento del montaggio del volume.

### Rivedere gli eventi

Astra Trident crea eventi per il nodo per identificare i servizi rilevati. Per rivedere questi eventi, eseguire:

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

### Esaminare i servizi rilevati

Astra Trident identifica i servizi abilitati per ciascun nodo sul nodo Trident CR. Per visualizzare i servizi rilevati, eseguire:

```
tridentctl get node -o wide -n <Trident namespace>
```

## Volumi NFS

Installa gli strumenti NFS utilizzando i comandi del tuo sistema operativo. Assicurarsi che il servizio NFS venga avviato durante l'avvio.

### RHEL 8+

```
sudo yum install -y nfs-utils
```

### Ubuntu

```
sudo apt-get install -y nfs-common
```



Riavviare i nodi di lavoro dopo aver installato gli strumenti NFS per evitare errori durante il collegamento dei volumi ai container.

## Volumi iSCSI

Astra Trident può stabilire automaticamente una sessione iSCSI, eseguire la scansione delle LUN e rilevare i dispositivi multipath, formattarli e montarli su un pod.

### Funzionalità di riparazione automatica di iSCSI

Per i sistemi ONTAP, Astra Trident esegue la riparazione automatica di iSCSI ogni cinque minuti per:

1. **Identificare** lo stato della sessione iSCSI desiderato e lo stato della sessione iSCSI corrente.
2. **Confrontare** lo stato desiderato con quello corrente per identificare le riparazioni necessarie. Astra Trident determina le priorità di riparazione e quando anticipare le riparazioni.
3. **Eseguire le riparazioni** necessarie per riportare lo stato della sessione iSCSI corrente allo stato della sessione iSCSI desiderato.



I registri delle attività di riparazione automatica si trovano in `trident-main` Container sul rispettivo pod `Demonset`. Per visualizzare i registri, è necessario aver impostato `debug "True"` durante l'installazione di Astra Trident.

Le funzionalità di riparazione automatica iSCSI di Astra Trident possono contribuire a prevenire:

- Sessioni iSCSI obsolete o non funzionanti che potrebbero verificarsi dopo un problema di connettività di rete. In caso di sessione obsoleta, Astra Trident attende sette minuti prima di disconnettersi per ristabilire la connessione con un portale.



Ad esempio, se i segreti CHAP sono stati ruotati sul controller di storage e la rete perde la connettività, i vecchi segreti CHAP (*stale*) potrebbero persistere. L'autoriparazione è in grado di riconoscerlo e ristabilire automaticamente la sessione per applicare i segreti CHAP aggiornati.

- Sessioni iSCSI mancanti

- LUN mancanti

## Installare gli strumenti iSCSI

Installare gli strumenti iSCSI utilizzando i comandi del sistema operativo.

### Prima di iniziare

- Ogni nodo del cluster Kubernetes deve avere un IQN univoco. **Questo è un prerequisito necessario.**
- Se si utilizza RHCOS versione 4.5 o successiva, o un'altra distribuzione Linux compatibile con RHEL, con `solidfire-san` Driver ed Element OS 12.5 o versioni precedenti, assicurarsi che l'algoritmo di autenticazione CHAP sia impostato su MD5 in `/etc/iscsi/iscsid.conf`. Gli algoritmi CHAP conformi a FIPS sicuri SHA1, SHA-256 e SHA3-256 sono disponibili con Element 12.7.

```
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, specificare `discard` MountOption in StorageClass per eseguire la rigenerazione dello spazio inline. Vedere ["Documentazione RedHat"](#).

## RHEL 8+

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 --find_multipaths n
```



Assicurarsi `etc/multipath.conf` contiene `find_multipaths no` sotto `defaults`.

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
```

## Ubuntu

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 no  
}  
EOF  
sudo systemctl enable --now multipath-tools.service  
sudo service multipath-tools restart
```



Assicurarsi `etc/multipath.conf` contiene `find_multipaths no` sotto `defaults`.

### 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.



Riavviare i nodi di lavoro dopo aver installato gli strumenti iSCSI per evitare errori durante il collegamento dei volumi ai container.

## Volumi NVMe/TCP

Installa gli strumenti NVMe utilizzando i comandi del tuo sistema operativo.



- NVMe richiede RHEL 9 o versione successiva.
- Se la versione del kernel del nodo Kubernetes è troppo vecchia o se il pacchetto NVMe non è disponibile per la versione del kernel in uso, potrebbe essere necessario aggiornare la versione del kernel del nodo a una versione con il pacchetto NVMe.

### RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

### Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

## Verificare l'installazione

Dopo l'installazione, verificare che ogni nodo nel cluster Kubernetes disponga di un NQN univoco utilizzando il comando:

```
cat /etc/nvme/hostnqn
```



Astra Trident modifica l' `ctrl_device_tmo` Value per garantire che NVMe non si arrenda sul percorso in caso di arresti. Non modificare questa impostazione.

# Configurare e gestire i backend

## 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 offre automaticamente pool di storage da backend che soddisfano i requisiti definiti da una classe di storage. Scopri come configurare il back-end per il tuo sistema storage.

- ["Configurare un backend Azure NetApp Files"](#)
- ["Configurare un Cloud Volumes Service per il backend della piattaforma cloud Google"](#)
- ["Configurare un backend NetApp HCI o SolidFire"](#)
- ["Configurare un backend con driver NAS ONTAP o Cloud Volumes ONTAP"](#)



- "Configurare un backend con i driver SAN ONTAP o Cloud Volumes ONTAP"
- "Utilizza Astra Trident con Amazon FSX per NetApp ONTAP"

## Azure NetApp Files

### Configurare un backend Azure NetApp Files

Puoi configurare Azure NetApp Files come back-end per Astra Trident. È possibile collegare volumi NFS e SMB utilizzando un backend Azure NetApp Files. Astra Trident supporta anche la gestione delle credenziali utilizzando identità gestite per i cluster Azure Kubernetes Services (AKS).

#### Dettagli del driver Azure NetApp Files

Astra Trident offre i seguenti driver di storage Azure NetApp Files per comunicare con il cluster. Le modalità di accesso supportate sono: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
azure-netapp-files	NFS PMI	Filesystem	RWO, ROX, RWX, RWOP	nfs, smb

### Considerazioni

- Il servizio Azure NetApp Files non supporta volumi inferiori a 100 GB. Astra Trident crea automaticamente volumi 100-GiB se viene richiesto un volume più piccolo.
- Astra Trident supporta volumi SMB montati su pod eseguiti solo su nodi Windows.

#### Identità gestite per AKS

Astra Trident supporta "identità gestite" Per i cluster di Azure Kubernetes Services. Per sfruttare al meglio la gestione semplificata delle credenziali offerta dalle identità gestite, è necessario disporre di:

- Un cluster Kubernetes implementato utilizzando AKS
- Identità gestite configurate sul cluster AKS kuBoost
- Astra Trident ha installato che include `cloudProvider` da specificare "Azure".

## Operatore Trident

Per installare Astra Trident usando l'operatore Trident, modifica `tridentorchestrator_cr.yaml` da impostare `cloudProvider` a "Azure". Ad esempio:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

## Timone

Nell'esempio seguente vengono installati i set Astra Trident `cloudProvider` In Azure utilizzando la variabile di ambiente `$CP`:

```
helm install trident trident-operator-23.10.0-custom.tgz --create
--namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

## `tridentctl`

Nell'esempio seguente viene installato Astra Trident imposta la `cloudProvider` contrassegna come Azure:

```
tridentctl install --cloud-provider="Azure" -n trident
```

## Prepararsi a configurare un backend Azure NetApp Files

Prima di poter configurare il backend Azure NetApp Files, è necessario assicurarsi che siano soddisfatti i seguenti requisiti.

### Prerequisiti per volumi NFS e SMB

Se si utilizza Azure NetApp Files per la prima volta o in una nuova posizione, è necessaria una configurazione iniziale per configurare Azure NetApp Files e creare un volume NFS. Fare riferimento a ["Azure: Configura Azure NetApp Files e crea un volume NFS"](#).

Per configurare e utilizzare un ["Azure NetApp Files"](#) back-end, sono necessari i seguenti elementi:



`subscriptionID`, `tenantID`, `clientID`, `location`, e `clientSecret` Sono opzionali quando si utilizzano identità gestite su un cluster AKS.

- Un pool di capacità. Fare riferimento a ["Microsoft: Creare un pool di capacità per Azure NetApp Files"](#).
- Una subnet delegata a Azure NetApp Files. Fare riferimento a ["Microsoft: Delegare una subnet a Azure NetApp Files"](#).
- `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:
  - Il ruolo di Proprietario o collaboratore ["Predefinito da Azure"](#).
  - R ["Ruolo di collaboratore personalizzato"](#) a livello di abbonamento (`assignableScopes`) Con le seguenti autorizzazioni limitate solo a quanto richiesto da Astra Trident. Dopo aver creato il ruolo personalizzato, ["Assegnare il ruolo utilizzando il portale Azure"](#).

## Ruolo collaboratore personalizzato

```
{
  "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited
permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.NetApp/netAppAccounts/capacityPools/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",
          "Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",
          "Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
```

```

ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/delete",
        "Microsoft.Features/features/read",
        "Microsoft.Features/operations/read",
        "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
        ],
        "notActions": [],
        "dataActions": [],
        "notDataActions": []
    }
  ]
}
}

```

- Azure location che ne contiene almeno uno **"subnet delegata"**. A partire da Trident 22.01, il location parametro è un campo obbligatorio al livello superiore del file di configurazione back-end. I valori di posizione specificati nei pool virtuali vengono ignorati.

### Requisiti aggiuntivi per i volumi SMB

Per creare un volume SMB, è necessario disporre di:

- Active Directory configurato e connesso a Azure NetApp Files. Fare riferimento a ["Microsoft: Creazione e gestione delle connessioni Active Directory per Azure NetApp Files"](#).
- Un cluster Kubernetes con un nodo controller Linux e almeno un nodo di lavoro Windows che esegue Windows Server 2019. Astra Trident supporta volumi SMB montati su pod eseguiti solo su nodi Windows.
- Almeno un segreto di Astra Trident contenente le credenziali di Active Directory in modo che Azure NetApp Files possa autenticarsi in Active Directory. Per generare un segreto smbcreds:

```

kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'

```

- Proxy CSI configurato come servizio Windows. Per configurare un `csi-proxy`, fare riferimento a ["GitHub: Proxy CSI"](#) oppure ["GitHub: Proxy CSI per Windows"](#) Per i nodi Kubernetes in esecuzione su Windows.

### Opzioni di configurazione back-end Azure NetApp Files ed esempi

Scopri le opzioni di configurazione di back-end NFS e SMB per Azure NetApp Files e

consulta gli esempi di configurazione.

### Opzioni di configurazione back-end

Astra Trident utilizza la configurazione backend (subnet, rete virtuale, livello di servizio e posizione) per creare volumi Azure NetApp Files su pool di capacità disponibili nel percorso richiesto e corrispondere al livello di servizio e alla subnet richiesti.



Astra Trident non supporta i pool di capacità QoS manuali.

I backend Azure NetApp Files forniscono queste opzioni di configurazione.

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  Opzionale quando le identità gestite sono abilitate su un cluster AKS.	
tenantID	L'ID tenant di una registrazione app  Opzionale quando le identità gestite sono abilitate su un cluster AKS.	
clientID	L'ID client di una registrazione dell'applicazione  Opzionale quando le identità gestite sono abilitate su un cluster AKS.	
clientSecret	Il segreto del client da una registrazione dell'applicazione  Opzionale quando le identità gestite sono abilitate su un cluster AKS.	
serviceLevel	Uno di Standard, Premium, o Ultra	"" (casuale)
location	Nome della posizione di Azure in cui verranno creati i nuovi volumi  Opzionale quando le identità gestite sono abilitate su un cluster AKS.	
resourceGroups	Elenco dei gruppi di risorse per filtrare le risorse rilevate	[] (nessun filtro)
netappAccounts	Elenco degli account NetApp per il filtraggio delle risorse rilevate	[] (nessun filtro)

Parametro	Descrizione	Predefinito
capacityPools	Elenco dei pool di capacità per filtrare le risorse rilevate	"" (nessun filtro, casuale)
virtualNetwork	Nome di una rete virtuale con una subnet delegata	""
subnet	Nome di una subnet delegata a. Microsoft.Netapp/volumes	""
networkFeatures	<p>Serie di funzionalità VNET per un volume, potrebbe essere Basic oppure Standard.</p> <p>Le funzioni di rete non sono disponibili in tutte le regioni e potrebbero essere abilitate in un abbonamento. Specificare networkFeatures se la funzionalità non è attivata, il provisioning del volume non viene eseguito correttamente.</p>	""
nfsMountOptions	<p>Controllo dettagliato delle opzioni di montaggio NFS.</p> <p>Ignorato per i volumi SMB.</p> <p>Per montare i volumi utilizzando NFS versione 4.1, include nfsvers=4 Nell'elenco delle opzioni di montaggio delimitate da virgole, scegliere NFS v4.1.</p> <p>Le opzioni di montaggio impostate in una definizione di classe di storage sovrascrivono le opzioni di montaggio impostate nella configurazione backend.</p>	"nfsvers=3"
limitVolumeSize	Il provisioning non riesce se le dimensioni del volume richiesto sono superiori a questo valore	"" (non applicato per impostazione predefinita)
debugTraceFlags	<p>Flag di debug da utilizzare per la risoluzione dei problemi. Esempio, <code>\{"api": false, "method": true, "discovery": true\}</code>.</p> <p>Non utilizzare questa opzione a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.</p>	nullo

Parametro	Descrizione	Predefinito
nasType	<p>Configurare la creazione di volumi NFS o SMB.</p> <p>Le opzioni sono <code>nfs</code>, <code>smb</code> o nullo. L'impostazione su Null consente di impostare i volumi NFS come predefiniti.</p>	nfs



Per ulteriori informazioni sulle funzioni di rete, fare riferimento a ["Configurare le funzionalità di rete per un volume Azure NetApp Files"](#).

### Autorizzazioni e risorse richieste

Se viene visualizzato l'errore "Nessun pool di capacità trovato" durante la creazione di un PVC, è probabile che la registrazione dell'applicazione non disponga delle autorizzazioni e delle risorse necessarie (subnet, rete virtuale, pool di capacità) associate. Se il debug è attivato, Astra Trident registra le risorse Azure rilevate al momento della creazione del backend. Verificare che venga utilizzato un ruolo appropriato.

I valori per `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork`, e `subnet` può essere specificato utilizzando nomi brevi o completi. Nella maggior parte dei casi, si consiglia di utilizzare nomi completi, in quanto i nomi brevi possono corrispondere a più risorse con lo stesso nome.

Il `resourceGroups`, `netappAccounts`, e `capacityPools` i valori sono filtri che limitano l'insieme di risorse rilevate a quelle disponibili per questo backend di storage e possono essere specificati in qualsiasi combinazione. I nomi pienamente qualificati seguono questo formato:

Tipo	Formato
Gruppo di risorse	<resource group>
Account NetApp	<resource group>/<netapp account>
Pool di capacità	<resource group>/<netapp account>/<capacity pool>
Rete virtuale	<resource group>/<virtual network>
Subnet	<resource group>/<virtual network>/<subnet>

### Provisioning di volumi

È possibile controllare il provisioning del volume predefinito specificando le seguenti opzioni in una sezione speciale del file di configurazione. Fare riferimento a [Configurazioni di esempio](#) per ulteriori informazioni.



Parametro	Descrizione	Predefinito
exportRule	Regole di esportazione per nuovi volumi.  exportRule Deve essere un elenco separato da virgole di qualsiasi combinazione di indirizzi IPv4 o subnet IPv4 nella notazione CIDR.  Ignorato per i volumi SMB.	"0.0.0.0/0"
snapshotDir	Controlla la visibilità della directory .snapshot	"falso"
size	La dimensione predefinita dei nuovi volumi	"100 G"
unixPermissions	Le autorizzazioni unix dei nuovi volumi (4 cifre ottali).  Ignorato per i volumi SMB.	"" (funzione di anteprima, richiede la whitelist nell'abbonamento)

### Configurazioni di esempio

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.

### 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 a Azure NetApp Files nel percorso configurato, e posiziona nuovi volumi in uno di tali pool e subnet in modo casuale. Perché `nasType` viene ommesso, il `nfs` Viene applicato il valore predefinito e il backend eseguirà il provisioning dei volumi NFS.

Questa configurazione è l'ideale se stai iniziando a utilizzare Azure NetApp Files e provando qualcosa, ma in pratica vorresti fornire un ulteriore ambito per i volumi da te forniti.

```
---
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
```

## Identità gestite per AKS

Questa configurazione di backend omette `subscriptionID`, `tenantID`, `clientID`, e `clientSecret`, che sono opzionali quando si utilizzano identità gestite.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools: ["ultra-pool"]
  resourceGroups: ["aks-ami-eastus-rg"]
  netappAccounts: ["smb-na"]
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
```

## Configurazione specifica del livello di servizio con filtri pool di capacità

Questa configurazione di back-end consente di posizionare i volumi in Azure `eastus` posizione in un `Ultra` pool di capacità. Astra Trident scopre automaticamente tutte le subnet delegate a Azure NetApp Files in tale posizione e posiziona 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: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
```

## 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: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
virtualNetwork: my-virtual-network
subnet: my-subnet
networkFeatures: Standard
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
  snapshotDir: 'true'
  size: 200Gi
  unixPermissions: '0777'
```

## Configurazione dei pool virtuali

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. Le etichette dei pool virtuali sono state utilizzate per differenziare i pool in base a performance.

```
---
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
resourceGroups:
- application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
- labels:
  performance: gold
  serviceLevel: Ultra
  capacityPools:
  - ultra-1
  - ultra-2
  networkFeatures: Standard
- labels:
  performance: silver
  serviceLevel: Premium
  capacityPools:
  - premium-1
- labels:
  performance: bronze
  serviceLevel: Standard
  capacityPools:
  - standard-1
  - standard-2
```

## Definizioni delle classi di storage

Quanto segue `StorageClass` le definizioni si riferiscono ai pool di storage sopra indicati.

## Definizioni di esempio con `parameter.selector` campo

Utilizzo di `parameter.selector` è possibile specificare per ciascuno `StorageClass` il pool virtuale 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
```

## Definizioni di esempio per volumi SMB

Utilizzo di `nasType`, `node-stage-secret-name`, e `node-stage-secret-namespace`, È possibile specificare un volume SMB e fornire le credenziali Active Directory richieste.

## Configurazione di base sullo spazio dei nomi predefinito

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

## Utilizzo di segreti diversi per spazio dei nomi

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

## Utilizzo di segreti diversi per volume

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```



`nasType: smb` Filtri per pool che supportano volumi SMB. `nasType: nfs` oppure `nasType: null` Filtri per i pool NFS.

### Creare il backend

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 Cloud Volumes Service per il backend di Google Cloud

Scopri come configurare NetApp Cloud Volumes Service per Google Cloud come backend per la tua installazione Astra Trident utilizzando le configurazioni di esempio fornite.

### Dettagli del driver di Google Cloud

Astra Trident offre a. `gcp-cvs` driver per comunicare con il quadro strumenti. Le modalità di accesso supportate sono: *ReadWriteOnce (RWO)*, *ReadOnlyMany (ROX)*, *ReadWriteMany (RWX)*, *ReadWriteOncePod (RWOP)*.

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
<code>gcp-cvs</code>	NFS	Filesystem	RWO, ROX, RWX, RWOP	<code>nfs</code>

### Scopri di più sul supporto di Astra Trident per Cloud Volumes Service per Google Cloud

Astra Trident può creare volumi Cloud Volumes Service in uno dei due "tipi di servizio":

- **CVS-Performance:** Il tipo di servizio Astra Trident predefinito. Questo tipo di servizio ottimizzato per le performance è più adatto per i carichi di lavoro di produzione che apprezzano le performance. Il tipo di servizio CVS-Performance è un'opzione hardware che supporta volumi con una dimensione minima di 100 GiB. È possibile scegliere tra "tre livelli di servizio":
  - `standard`
  - `premium`
  - `extreme`
- **CVS:** Il tipo di servizio CVS offre un'elevata disponibilità zonale con livelli di performance da limitati a moderati. Il tipo di servizio CVS è un'opzione software che utilizza pool di storage per supportare volumi di

dimensioni pari a 1 GiB. Il pool di storage può contenere fino a 50 volumi in cui tutti i volumi condividono la capacità e le performance del pool. È possibile scegliere tra "due livelli di servizio":

- `standardsw`
- `zoneredundantstandardsw`

## 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 Cloud Volumes Service
- Numero di progetto dell'account Google Cloud
- Account di servizio Google Cloud con `netappcloudvolumes.admin` ruolo
- File delle chiavi API per l'account Cloud Volumes Service

## Opzioni di configurazione back-end

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.

Parametro	Descrizione	Predefinito
<code>version</code>		Sempre 1
<code>storageDriverName</code>	Nome del driver di storage	"gcp-cvs"
<code>backendName</code>	Nome personalizzato o backend dello storage	Nome del driver + "_" + parte della chiave API
<code>storageClass</code>	Parametro facoltativo utilizzato per specificare il tipo di servizio CVS.  Utilizzare <code>software</code> Per selezionare il tipo di servizio CVS. In caso contrario, Astra Trident presuppone il tipo di servizio CVS-Performance ( <code>hardware</code> ).	
<code>storagePools</code>	Solo tipo di servizio CVS. Parametro facoltativo utilizzato per specificare i pool di storage per la creazione di volumi.	
<code>projectNumber</code>	Numero di progetto dell'account Google Cloud. Il valore si trova nella home page del portale Google Cloud.	
<code>hostProjectNumber</code>	Necessario se si utilizza una rete VPC condivisa. In questo scenario, <code>projectNumber</code> è il progetto di servizio, e <code>hostProjectNumber</code> è il progetto host.	
<code>apiRegion</code>	La regione di Google Cloud in cui Astra Trident crea volumi Cloud Volumes Service. Quando si creano cluster Kubernetes con più aree, i volumi creati in un <code>apiRegion</code> Può essere utilizzato nei carichi di lavoro pianificati su nodi in più aree di Google Cloud.  Il traffico interregionale comporta un costo aggiuntivo.	



Parametro	Descrizione	Predefinito
apiKey	<p>Chiave API per l'account del servizio Google Cloud con <code>netappcloudvolumes.admin</code> ruolo.</p> <p>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).</p>	
proxyURL	<p>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.</p> <p>Per un proxy HTTPS, la convalida del certificato viene ignorata per consentire l'utilizzo di certificati autofirmati nel server proxy.</p> <p>I server proxy con autenticazione abilitata non sono supportati.</p>	
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	<p>Livello di servizio CVS-Performance o CVS per i nuovi volumi.</p> <p>I valori CVS-Performance sono <code>standard</code>, <code>premium</code>, o <code>extreme</code>.</p> <p>I valori CVS sono <code>standardsw</code> oppure <code>zoneredundantstandardsw</code>.</p>	<p>CVS-Performance (prestazioni CVS) è "standard".</p> <p>Il valore predefinito di CVS è "standardsw".</p>
network	Rete Google Cloud utilizzata per i volumi Cloud Volumes Service.	"predefinito"
debugTraceFlags	<p>Flag di debug da utilizzare per la risoluzione dei problemi. Esempio, <code>\{"api":false, "method":true\}</code>.</p> <p>Non utilizzare questa opzione a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.</p>	nullo
allowedTopologies	<p>Per abilitare l'accesso multi-regione, la definizione <code>StorageClass</code> per <code>allowedTopologies</code> deve includere tutte le regioni.</p> <p>Ad esempio:</p> <ul style="list-style-type: none"> <li>- <code>key: topology.kubernetes.io/region</code></li> <li><code>values:</code></li> <li>- <code>us-east1</code></li> <li>- <code>europa-west1</code></li> </ul>	

## Opzioni di provisioning dei volumi

È possibile controllare il provisioning del volume predefinito in `defaults` del file di configurazione.

Parametro	Descrizione	Predefinito
<code>exportRule</code>	Le regole di esportazione per i nuovi volumi. Deve essere un elenco separato da virgole di qualsiasi combinazione di indirizzi IPv4 o subnet IPv4 nella notazione CIDR.	"0.0.0.0/0"
<code>snapshotDir</code>	Accesso a <code>.snapshot directory</code>	"falso"
<code>snapshotReserve</code>	Percentuale di volume riservato agli snapshot	"" (accettare CVS come valore predefinito 0)
<code>size</code>	Le dimensioni dei nuovi volumi.  Performance CVS minima: 100 GiB.  CVS minimo: 1 GiB.	Per impostazione predefinita, il tipo di servizio CVS-Performance è "100GiB".  Il tipo di servizio CVS non imposta un valore predefinito, ma richiede un minimo di 1 GiB.

## Esempi di tipo di servizio CVS-Performance

I seguenti esempi forniscono configurazioni di esempio per il tipo di servizio CVS-Performance.



```
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 livello di servizio

In questo esempio vengono illustrate le opzioni di configurazione back-end, inclusi il livello di servizio e i valori predefiniti del volume.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  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
```



```

znHczZsrtrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
XsYg6gyxy4zq7OlwWgLwGa==
-----END PRIVATE KEY-----
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
```

### **Definizioni delle classi di storage**

Le seguenti definizioni di StorageClass si applicano all'esempio di configurazione del pool virtuale. Utilizzo di `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.

## Esempio di classe di storage

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

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

- Il primo StorageClass (`cvs-extreme-extra-protection`) viene mappato al primo pool 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 virtuale è selezionato e garantisce che il requisito di riserva snapshot sia soddisfatto.

### Esempi di tipo di servizio CVS

I seguenti esempi forniscono configurazioni di esempio per il tipo di servizio CVS.



```
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  
serviceLevel: standardsw
```

## Esempio 2: Configurazione del pool di storage

Questo esempio di configurazione di backend utilizza `storagePools` per configurare un pool di storage.

```
---
version: 1
storageDriverName: gcp-cvs
backendName: gcp-std-so-with-pool
projectNumber: '531265380079'
apiRegion: europe-west1
apiKey:
  type: service_account
  project_id: cloud-native-data
  private_key_id: "<id_value>"
  private_key: |-
    -----BEGIN PRIVATE KEY-----
    MIIEvAIBADANBgkqhkiG9w0BAQEFAASCbKYwggSiAgEAAoIBAQDaT+Oui9FBAw19
    L1AGEkrYU5xd9K5NlO5jMkIFND5wCD+Nv+jd1GvtFRLaLK5RvXyF5wzvztmODNS+
    qtScpQ+5cFpQkuGtv9U9+N6qtuVYYO3b504Kp5CtqVPJCgMJaK2j8pZTIqUiMum/
    5/Y9oTbZrjAHSMsgJm2nHzFq2X0rqVMAhghI6ATm4DOuWx8XGWKGTGIPlc0qPqJlqS
    LLaWOH4VIZQZCAyW5IUp9CAmwqHgdG0uhFNfCgMmED6PBUvVLsLvcq86X+QSWR9k
    ETqElj/sGCenPF7ti1DhGBFafd9hPnxg9PZY29ArEZwY9G/ZjZQX7WPgs0VvxiNR
    DxZRC3GXAgMBAECggEACn5c59bG/qnVEVI1CwMAalM5M2z09JFh1L1ljKwntNPj
    Vilw2eTW2+UE7HbJru/S7KQgA5Dnn9kvCraEahPRuddUMrD0vG4kTl/IODV6uFuk
    Y0sZfbqd4jMUQ21smvGsqFzwloYWS5qzO1W83ivXH/HW/iqkmY2eW+EPRS/hwSSu
    SscR+SojI7PB0BWSJhlV4yqYf3vcd/D95el2CVHfRCkL85DKumeZ+yHEnpiXGZAE
    t8xSs4a500Pm6NHhevCw2a/UQ95/foXNUR450HtbjieJo5o+FF6EYZQGfU2ZHZO8
    37FBKuaJkdGW5xqaI9TL7aqkGkFMF4F2qvOZM+vy8QKBgQD4oVuOkJD1hkTHP86W
    esFlw1kpWyJR9ZA7LI0g/rVpslnX+XdDq0WQf4umdLNau5hYEH9LU6ZSGs1Xk3/B
    NHwR6OXFuqEKNiu83d0zSlHhTy7PZpOZdj5a/vVvQfPDMz7OvsqLRd7YCAbdzuQ0
    +Ahq0Ztwvg0HQ64hdW0ukpYRRwKBgQDgyHj98oqsw0YuIa+pP1yS0pPwLmjwKyNm
    /HayzCp+Qjiiyy7Tzg8AUqlH1Ou83Xbv428jvg7kDh07PCCKFq+mMmfqHmTpb0Maq
    KpKnZg4ipsqPlyHNNEOrmcailXbwIhCLewMqMrggUiLOmCw4PscL5nK+4GKu2XE1
    jLqjWAZFMQKBgFHkQ9XXRAJ1kR3XpGHoGN890pZOkCVSrqju6aUef/5KY1FCt8ew
    F/+aIxM2iQsvmWQYOvVCnhuY/F2GFAQ7d0om3decuwI0CX/xy7PjHMkLXa2uaZs4
    WR17sLduj62RqXRLX0c0QkwBiNFyHbRcpdkZJQujbyMhBa+7j7SxT4BtAoGAWMWT
    UucocRXZm/pdvz9wteNH3YDwnJLMxm1KC06qMXbBoYrliY4sm3ywJWMC+iCd/H8A
    Gecxd/xVu5mA2L2N3KMq18Zhz8Th0G5DwKyDRJgOQ0Q46yuNXOoYEjlo4Wjyk8Me
    +tlQ8iK98E0UmZnhTgfSpSNElzbz2AqnzQ3MN9uECgYAqdvdpVnKGFvdtZ2DjyMoJ
    E89UIC41WjjJGmHsd8W65+3X0RwMzKMT6aZc5tK9J5dHvmWIETnbM+1TImdbBFga
    NWOC6f3r2xbGXHhaWS1+nobpTuvlo56ZRJVvV71FMsiddzMuHH8pxfgNjemwA4P
    ThDHcejv035NNV6Kyo00tA==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@cloud-native-
  data.iam.gserviceaccount.com
  client_id: '107071413297115343396'
```

```
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%40cloud-native-data.iam.gserviceaccount.com
storageClass: software
zone: europe-west1-b
network: default
storagePools:
- 1bc7f380-3314-6005-45e9-c7dc8c2d7509
serviceLevel: Standardsw
```

## 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 di elementi con la tua installazione Astra Trident.

### Dettagli driver elemento

Astra Trident offre al `solidfire-san` driver di archiviazione per comunicare con il cluster. Le modalità di accesso supportate sono: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Il `solidfire-san` il driver di archiviazione supporta 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, RWOP	Nessun filesystem. Dispositivo a blocchi raw.
solidfire-san	ISCSI	Filesystem	RWO, RWOP	xfs, ext3, ext4

## Prima di iniziare

Prima di creare un backend elemento, è necessario quanto segue.

- 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"](#).

## 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.	""
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. Astra Trident utilizza il protocollo CHAP.	vero
AccessGroups	Elenco degli ID del gruppo di accesso da utilizzare	Trova l'ID di un gruppo di accesso denominato "tridente"
Types	Specifiche QoS	



Parametro	Descrizione	Predefinito
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.

### 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 virtuali

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

Astra Trident copia le etichette presenti su un pool di storage nel LUN dello storage back-end al momento del provisioning. Per comodità, gli amministratori dello storage possono definire le etichette per ogni pool virtuale e raggruppare i volumi per etichetta.

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 virtuali sono definiti in `storage` sezione. In questo esempio, alcuni pool di storage impostano il proprio tipo e alcuni pool sovrascrivono i valori predefiniti impostati in precedenza.

```
---
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 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 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 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"](#)

## Driver SAN ONTAP

### Panoramica del driver SAN ONTAP

Informazioni sulla configurazione di un backend ONTAP con driver SAN ONTAP e Cloud Volumes ONTAP.

### Dettagli del driver SAN ONTAP

Astra Trident offre i seguenti driver per lo storage SAN per comunicare con il cluster ONTAP. Le modalità di accesso supportate sono: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).



Se stai utilizzando Astra Control per protezione, recovery e mobilità, leggi [Compatibilità driver Astra Control](#).

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
ontap-san	ISCSI	Blocco	RWO, ROX, RWX, RWOP	Nessun file system; dispositivo a blocchi raw
ontap-san	ISCSI	Filesystem	RWO, RWOP  ROX e RWX non sono disponibili in modalità Volume filesystem.	xfv, ext3, ext4
ontap-san	NVMe/TCP  Fare riferimento a. <a href="#">Considerazioni aggiuntive su NVMe/TCP</a> .	Blocco	RWO, ROX, RWX, RWOP	Nessun file system; dispositivo a blocchi raw

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
ontap-san	NVMe/TCP  Fare riferimento a. <a href="#">Considerazioni aggiuntive su NVMe/TCP.</a>	Filesystem	RWO, RWOP  ROX e RWX non sono disponibili in modalità Volume filesystem.	xfs, ext3, ext4
ontap-san-economy	ISCSI	Blocco	RWO, ROX, RWX, RWOP	Nessun file system; dispositivo a blocchi raw
ontap-san-economy	ISCSI	Filesystem	RWO, RWOP  ROX e RWX non sono disponibili in modalità Volume filesystem.	xfs, ext3, ext4

### Compatibilità driver Astra Control

Astra Control offre protezione perfetta, disaster recovery e mobilità (spostamento di volumi tra cluster Kubernetes) per i volumi creati con `ontap-nas`, `ontap-nas-flexgroup`, e `ontap-san` driver. Vedere ["Prerequisiti per la replica di Astra Control"](#) per ulteriori informazioni.



- Utilizzare `ontap-san-economy` solo se si prevede che il conteggio dell'utilizzo persistente del volume sia superiore a. ["Limiti di volume ONTAP supportati"](#).
- Utilizzare `ontap-nas-economy` solo se si prevede che il conteggio dell'utilizzo persistente del volume sia superiore a. ["Limiti di volume ONTAP supportati"](#) e a. `ontap-san-economy` impossibile utilizzare il driver.
- Non utilizzare `ontap-nas-economy` se prevedete la necessità di protezione dei dati, disaster recovery o mobilità.

### 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.

Sebbene sia possibile creare un ruolo più restrittivo all'interno di ONTAP che un driver Trident può utilizzare, non lo consigliamo. La maggior parte delle nuove release di Trident chiamerà API aggiuntive che dovrebbero essere considerate, rendendo gli aggiornamenti difficili e soggetti a errori.

### Considerazioni aggiuntive su NVMe/TCP

Astra Trident supporta il protocollo non-volatile memory express (NVMe) utilizzando il `ontap-san` driver che include:

- IPv6
- Snapshot e cloni di volumi NVMe
- Ridimensionamento di un volume NVMe
- Importare un volume NVMe creato al di fuori di Astra Trident in modo che il suo ciclo di vita possa essere gestito da Astra Trident
- Multipath nativo NVMe
- Arresto anomalo o anomalo dei K8s nodi (23,10)

Astra Trident non supporta:

- DH-HMAC-CHAP supportato nativamente da NVMe
- Multipathing DM (Device mapper)
- Crittografia LUKS

### Prepararsi a configurare il backend con i driver SAN ONTAP

Comprendere i requisiti e le opzioni di autenticazione per la configurazione di un backend ONTAP con i driver SAN ONTAP.

#### Requisiti

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. Fare riferimento a ["Preparare il nodo di lavoro"](#) per ulteriori informazioni.

#### Autenticare il backend ONTAP

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).

È possibile aggiornare i backend esistenti per passare da un metodo basato su credenziali a un metodo

basato su certificato. Tuttavia, è supportato un solo metodo di autenticazione alla volta. Per passare a un metodo di autenticazione diverso, è necessario rimuovere il metodo esistente dalla configurazione di back-end.



Se si tenta di fornire **credenziali e certificati**, la creazione del backend non riesce e viene visualizzato un errore che indica che nel file di configurazione sono stati forniti più metodi di autenticazione.

### 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:

#### YAML

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

#### JSON

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

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 o l'aggiornamento 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",
  "svm": "vserver_test",
  "clientCertificate": "Faaaakkkkeeee...Vaaalllluuueeeee",
  "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, è necessario rimuovere il metodo di autenticazione esistente e aggiungere il nuovo metodo di autenticazione. Quindi, utilizzare il file `backend.json` aggiornato contenente i parametri necessari per l'esecuzione `tridentctl backend update`.

```
cat cert-backend-updated.json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "SanBackend",
  "managementLIF": "1.2.3.4",
  "svm": "vserver_test",
  "username": "vsadmin",
  "password": "password",
  "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.

#### 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 sicurezza dell'iniziatore predefinito della SVM su CHAP bidirezionale e imposta il nome utente e i segreti dal 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: password
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 iniziatore predefinito della SVM è nessuno (impostato per impostazione predefinita) e nel volume non sono già presenti LUN preesistenti, 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. In questo modo, l'accesso ai LUN già presenti nella SVM non è 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).

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": "password",
  "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 ed esempi di configurazione DELLA SAN ONTAP

Scopri come creare e utilizzare i driver SAN ONTAP con la tua installazione Astra Trident. In questa sezione vengono forniti esempi di configurazione backend e dettagli per la mappatura dei 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	<p>Indirizzo IP di un cluster o di una LIF di gestione SVM.</p> <p>È possibile specificare un nome di dominio completo (FQDN).</p> <p>Può essere impostato per utilizzare gli indirizzi IPv6 se Astra Trident è stato installato utilizzando il flag IPv6. Gli indirizzi IPv6 devono essere definiti tra parentesi quadre, ad esempio [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].</p> <p>Per uno switchover perfetto del MetroCluster, consulta il <a href="#">Esempio MetroCluster</a>.</p>	"10.0.0.1", "[2001:1234:abcd::fefe]"
dataLIF	<p>Indirizzo IP del protocollo LIF.</p> <p><b>Non specificare iSCSI.</b> utilizza Astra Trident "<a href="#">Mappa LUN selettiva ONTAP</a>" Per scoprire i LIF iSCSI necessari per stabilire una sessione multi-percorso. Viene generato un avviso se dataLIF è esplicitamente definito.</p> <p><b>Ometti per MetroCluster.</b> vedere la <a href="#">Esempio MetroCluster</a>.</p>	Derivato dalla SVM
svm	<p>Macchina virtuale per lo storage da utilizzare</p> <p><b>Ometti per MetroCluster.</b> vedere la <a href="#">Esempio MetroCluster</a>.</p>	Derivato se un SVM managementLIF è specificato
useCHAP	<p>Utilizzare CHAP per autenticare iSCSI per i driver SAN ONTAP [booleano].</p> <p>Impostare su true Affinché Astra Trident configuri e utilizzi CHAP bidirezionale come autenticazione predefinita per la SVM fornita nel backend. Fare riferimento a. "<a href="#">Prepararsi a configurare il backend con i driver SAN ONTAP</a>" per ulteriori informazioni.</p>	false
chapInitiatorSecret	Segreto iniziatore CHAP. Necessario se useCHAP=true	""
labels	Set di etichette arbitrarie formattate con JSON da applicare sui volumi	""

Parametro	Descrizione	Predefinito
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	""
trustedCACertificate	Valore codificato in base64 del certificato CA attendibile. Opzionale. Utilizzato per l'autenticazione basata su certificato.	""
username	Nome utente necessario per comunicare con il cluster ONTAP. Utilizzato per l'autenticazione basata su credenziali.	""
password	Password necessaria per comunicare con il cluster ONTAP. Utilizzato per l'autenticazione basata su credenziali.	""
svm	Macchina virtuale per lo storage da utilizzare	Derivato se un SVM managementLIF è specificato
storagePrefix	Prefisso utilizzato per il provisioning di nuovi volumi nella SVM.  Non può essere modificato in seguito. Per aggiornare questo parametro, è necessario creare un nuovo backend.	trident
limitAggregateUsage	Il provisioning non riesce se l'utilizzo è superiore a questa percentuale.  Se si utilizza un backend Amazon FSX per NetApp ONTAP, non specificare limitAggregateUsage. Il fornito fsxadmin e vsadmin Non includere le autorizzazioni necessarie per recuperare l'utilizzo aggregato e limitarlo utilizzando Astra Trident.	"" (non applicato per impostazione predefinita)
limitVolumeSize	Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore.  Limita inoltre le dimensioni massime dei volumi gestiti per qtree e LUN.	"" (non applicato per impostazione predefinita)
lunsPerFlexvol	LUN massimi per FlexVol, devono essere compresi nell'intervallo [50, 200]	100

Parametro	Descrizione	Predefinito
debugTraceFlags	<p>Flag di debug da utilizzare per la risoluzione dei problemi. Esempio, {"api":false, "method":true}</p> <p>Non utilizzare a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.</p>	null
useREST	<p>Parametro booleano per l'utilizzo delle API REST di ONTAP. <b>Anteprima tecnica</b></p> <p>useREST viene fornito come <b>anteprima tecnica</b> 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 9.11.1 e versioni successive. Inoltre, il ruolo di accesso ONTAP utilizzato deve avere accesso a. <code>ontap</code> applicazione. Ciò è soddisfatto dal predefinito <code>vsadmin</code> e <code>cluster-admin</code> ruoli.</p> <p>useREST Non è supportato con MetroCluster.</p> <p>useREST È pienamente qualificato per NVMe/TCP.</p>	false
sanType	Utilizzare per selezionare <code>iscsi</code> Per iSCSI o <code>nvme</code> Per NVMe/TCP.	iscsi se vuoto

#### Opzioni di configurazione back-end per il provisioning dei volumi

È possibile controllare il provisioning predefinito utilizzando queste opzioni in `defaults` 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 prenotazione dello spazio; "nessuno" (sottile) o "volume" (spesso)	"nessuno"
snapshotPolicy	Policy di Snapshot da utilizzare	"nessuno"



Parametro	Descrizione	Predefinito
qosPolicy	<p>Gruppo di criteri QoS da assegnare per i volumi creati. Scegliere tra qosPolicy o adaptiveQosPolicy per pool di storage/backend.</p> <p>L'utilizzo di gruppi di policy QoS con Astra Trident richiede ONTAP 9.8 o versione successiva. Si consiglia di utilizzare un gruppo di policy QoS non condiviso e di assicurarsi che il gruppo di policy venga 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.</p>	""
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"
encryption	<p>Abilitare NetApp Volume Encryption (NVE) sul nuovo volume; il valore predefinito è <code>false</code>. NVE deve essere concesso in licenza e abilitato sul cluster per utilizzare questa opzione.</p> <p>Se NAE è attivato sul backend, tutti i volumi forniti in Astra Trident saranno abilitati per NAE.</p> <p>Per ulteriori informazioni, fare riferimento a: <a href="#">"Come funziona Astra Trident con NVE e NAE"</a>.</p>	"falso"
luksEncryption	<p>Attivare la crittografia LUKS. Fare riferimento a: <a href="#">"Utilizzo di Linux Unified Key Setup (LUKS)"</a>.</p> <p>La crittografia LUKS non è supportata per NVMe/TCP.</p>	""
securityStyle	Stile di sicurezza per nuovi volumi	unix
tieringPolicy	Criterio di tiering da utilizzare "nessuno"	"Solo Snapshot" per la configurazione SVM-DR pre-ONTAP 9,5

## Esempi di provisioning di volumi

Ecco un esempio con i valori predefiniti definiti:

```

---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
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.

### Esempio DI SAN ONTAP

Si tratta di una configurazione di base che utilizza `ontap-san` driver.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

### Esempio di economia SAN ONTAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
username: vsadmin
password: <password>
```

## Esempio MetroCluster

È possibile configurare il backend per evitare di dover aggiornare manualmente la definizione del backend dopo lo switchover e lo switchback durante ["Replica e recovery di SVM"](#).

Per uno switchover e uno switchback perfetto, specifica la SVM utilizzando `managementLIF` e omettere `dataLIF` e `svm` parametri. Ad esempio:

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

## Esempio di autenticazione basata su certificato

In questo esempio di configurazione di base `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
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

## Esempi CHAP bidirezionali

Questi esempi creano un backend con `useCHAP` impostare su `true`.

### Esempio di SAN ONTAP CHAP

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
```

### Esempio di ONTAP SAN economy CHAP

```
---
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
username: vsadmin
password: <password>
```

## Esempio NVMe/TCP

Devi disporre di una SVM configurata con NVMe sul back-end ONTAP. Si tratta di una configurazione backend di base per NVMe/TCP.

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

## Esempi di backend con pool virtuali

In questi file di definizione back-end di esempio, 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 virtuali sono definiti nella sezione `storage`.

Astra Trident imposta le etichette di provisioning nel campo "commenti". I commenti vengono impostati su FlexVol. Astra Trident copia tutte le etichette presenti su un pool virtuale nel volume di storage al momento del provisioning. Per comodità, gli amministratori dello storage possono definire le etichette per ogni pool virtuale e raggruppare i volumi per etichetta.

In questi esempi, alcuni dei pool di storage sono impostati in modo personalizzato `spaceReserve`, `spaceAllocation`, e `encryption` e alcuni pool sovrascrivono i valori predefiniti.

**Esempio DI SAN ONTAP**



```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
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'
```



## Esempio di economia SAN ONTAP

```
---
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
username: vsadmin
password: <password>
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'
- labels:
  department: legal
  creditpoints: '5000'
  zone: us_east_1c
```

```
defaults:
  spaceAllocation: 'true'
  encryption: 'false'
```

## Esempio NVMe/TCP

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: 'false'
  encryption: 'true'
storage:
- labels:
  app: testApp
  cost: '20'
  defaults:
    spaceAllocation: 'false'
    encryption: 'false'
```

## Mappare i backend in StorageClasses

Le seguenti definizioni di StorageClass fanno riferimento a [Esempi di backend con pool virtuali](#). 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 `protection-gold` StorageClass verrà mappato al primo pool virtuale in `ontap-san` back-end. Questo è l'unico pool che offre una protezione di livello gold.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- Il `protection-not-gold` StorageClass eseguirà il mapping al secondo e al terzo pool virtuale in `ontap-san` back-end. Questi sono gli unici pool che offrono un livello di protezione diverso dall'oro.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- Il `app-mysqldb` StorageClass eseguirà il mapping al terzo pool virtuale in `ontap-san-economy` back-end. Questo è l'unico pool che offre la configurazione del pool di storage per l'applicazione di tipo `mysqldb`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- Il `protection-silver-creditpoints-20k` StorageClass eseguirà il mapping al secondo pool virtuale in `ontap-san` back-end. Questo è l'unico pool che offre una protezione di livello Silver e 20000 punti di credito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- Il `creditpoints-5k` StorageClass eseguirà il mapping al terzo pool virtuale in `ontap-san` il back-end e il quarto pool virtuale in `ontap-san-economy` back-end. Queste sono le uniche offerte di pool con 5000 punti di credito.

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"

```

- Il my-test-app-sc StorageClass verrà mappato su testAPP pool virtuale in ontap-san conducente con sanType: nvme. Si tratta dell'unica offerta di piscina testApp.

```

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"

```

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

## Driver NAS ONTAP

### Panoramica del driver NAS ONTAP

Informazioni sulla configurazione di un backend ONTAP con driver NAS ONTAP e Cloud Volumes ONTAP.

#### Dettagli del driver NAS ONTAP

Astra Trident offre i seguenti driver di storage NAS per comunicare con il cluster ONTAP. Le modalità di accesso supportate sono: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).



Se stai utilizzando Astra Control per protezione, recovery e mobilità, leggi [Compatibilità driver Astra Control](#).

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
ontap-nas	NFS PMI	Filesystem	RWO, ROX, RWX, RWOP	"", nfs, smb

Driver	Protocollo	VolumeMode	Modalità di accesso supportate	File system supportati
ontap-nas-economy	NFS PMI	Filesystem	RWO, ROX, RWX, RWOP	"", nfs, smb
ontap-nas-flexgroup	NFS PMI	Filesystem	RWO, ROX, RWX, RWOP	"", nfs, smb

### Compatibilità driver Astra Control

Astra Control offre protezione perfetta, disaster recovery e mobilità (spostamento di volumi tra cluster Kubernetes) per i volumi creati con `ontap-nas`, `ontap-nas-flexgroup`, e `ontap-san` driver. Vedere ["Prerequisiti per la replica di Astra Control"](#) per ulteriori informazioni.



- Utilizzare `ontap-san-economy` solo se si prevede che il conteggio dell'utilizzo persistente del volume sia superiore a ["Limiti di volume ONTAP supportati"](#).
- Utilizzare `ontap-nas-economy` solo se si prevede che il conteggio dell'utilizzo persistente del volume sia superiore a ["Limiti di volume ONTAP supportati"](#) e a `ontap-san-economy` impossibile utilizzare il driver.
- Non utilizzare `ontap-nas-economy` se prevedete la necessità di protezione dei dati, disaster recovery o mobilità.

### 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.

Sebbene sia possibile creare un ruolo più restrittivo all'interno di ONTAP che un driver Trident può utilizzare, non lo consigliamo. La maggior parte delle nuove release di Trident chiamerà API aggiuntive che dovrebbero essere considerate, rendendo gli aggiornamenti difficili e soggetti a errori.

### Prepararsi a configurare un backend con i driver NAS ONTAP

Comprendere i requisiti, le opzioni di autenticazione e le policy di esportazione per la configurazione di un backend ONTAP con i driver NAS ONTAP.

## Requisiti

- Per tutti i backend ONTAP, Astra Trident richiede almeno un aggregato assegnato alla SVM.
- È possibile eseguire più di un driver e creare classi di storage che puntano 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.
- Astra Trident supporta volumi SMB montati su pod eseguiti solo su nodi Windows. Fare riferimento a [Preparatevi al provisioning dei volumi SMB](#) per ulteriori informazioni.

## Autenticare il backend ONTAP

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

- Basato sulle credenziali: Questa modalità richiede autorizzazioni sufficienti per il backend ONTAP. Si consiglia di utilizzare un account associato a 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: Questa modalità richiede un certificato installato sul backend affinché Astra Trident possa comunicare con un cluster ONTAP. 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).

È possibile aggiornare i backend esistenti per passare da un metodo basato su credenziali a un metodo basato su certificato. Tuttavia, è supportato un solo metodo di autenticazione alla volta. Per passare a un metodo di autenticazione diverso, è necessario rimuovere il metodo esistente dalla configurazione di back-end.



Se si tenta di fornire **credenziali e certificati**, la creazione del backend non riesce e viene visualizzato un errore che indica che nel file di configurazione sono stati forniti più metodi di autenticazione.

## 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:

## YAML

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

## JSON

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

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'update 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...Vaaalllluuueeee",
  "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, è necessario rimuovere il metodo di autenticazione esistente e aggiungere il nuovo metodo di autenticazione. Quindi, utilizzare il file backend.json aggiornato contenente i parametri necessari per l'esecuzione `tridentctl update backend`.

```

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": "password",
"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

Astra Trident permette di gestire in modo dinamico 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.



Non utilizzare NAT (Network Address Translation) quando si utilizzano criteri di esportazione dinamici. Con NAT, il controller di archiviazione rileva l'indirizzo NAT di frontend e non l'indirizzo host IP effettivo, pertanto l'accesso viene negato quando non viene trovata alcuna corrispondenza nelle regole di esportazione.

## Esempio

È necessario utilizzare due opzioni di configurazione. Ecco un esempio di definizione di backend:

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
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 creato in precedenza con una regola di esportazione che consenta il blocco CIDR del nodo (ad esempio il criterio di esportazione predefinito). Segui sempre le Best practice consigliate da NetApp per dedicare una SVM a 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>
      fileType: 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 verrà creato un nuovo criterio di esportazione denominato dopo l'UUID del backend e i volumi presenti sul backend utilizzeranno il

criterio di esportazione appena creato quando vengono nuovamente montati.



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.

### Preparatevi al provisioning dei volumi SMB

Con un po' di preparazione aggiuntiva, puoi eseguire il provisioning dei volumi SMB utilizzando `ontap-nas` driver.



Per creare un, è necessario configurare entrambi i protocolli NFS e SMB/CIFS su SVM `ontap-nas-economy` Volume SMB per ONTAP on-premise. La mancata configurazione di uno di questi protocolli causerà un errore nella creazione del volume SMB.

### Prima di iniziare

Prima di eseguire il provisioning di volumi SMB, è necessario disporre di quanto segue.

- Un cluster Kubernetes con un nodo controller Linux e almeno un nodo di lavoro Windows che esegue Windows Server 2019. Astra Trident supporta volumi SMB montati su pod eseguiti solo su nodi Windows.
- Almeno un segreto Astra Trident contenente le credenziali Active Directory. Per generare un segreto `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user  
--from-literal password='password'
```

- Proxy CSI configurato come servizio Windows. Per configurare un `csi-proxy`, fare riferimento a "[GitHub: Proxy CSI](#)" oppure "[GitHub: Proxy CSI per Windows](#)" Per i nodi Kubernetes in esecuzione su Windows.

### Fasi

1. Per ONTAP on-premise, è possibile creare una condivisione SMB oppure Astra Trident ne può creare una per te.



Le condivisioni SMB sono richieste per Amazon FSX per ONTAP.

È possibile creare le condivisioni amministrative SMB in due modi utilizzando "[Console di gestione Microsoft](#)" Snap-in cartelle condivise o utilizzo dell'interfaccia CLI di ONTAP. Per creare le condivisioni SMB utilizzando la CLI ONTAP:

- a. Se necessario, creare la struttura del percorso di directory per la condivisione.

Il `vserver cifs share create` il comando controlla il percorso specificato nell'opzione `-path` durante la creazione della condivisione. Se il percorso specificato non esiste, il comando non riesce.

- b. Creare una condivisione SMB associata alla SVM specificata:

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

c. Verificare che la condivisione sia stata creata:

```
vserver cifs share show -share-name share_name
```



Fare riferimento a ["Creare una condivisione SMB"](#) per informazioni dettagliate.

2. Quando si crea il backend, è necessario configurare quanto segue per specificare i volumi SMB. Per tutte le opzioni di configurazione backend FSX per ONTAP, fare riferimento a ["FSX per le opzioni di configurazione e gli esempi di ONTAP"](#).

Parametro	Descrizione	Esempio
smbShare	È possibile specificare una delle seguenti opzioni: Il nome di una condivisione SMB creata utilizzando la console di gestione Microsoft o l'interfaccia utente di ONTAP; un nome per consentire ad Astra Trident di creare la condivisione SMB; oppure è possibile lasciare vuoto il parametro per impedire l'accesso condiviso ai volumi.  Questo parametro è facoltativo per ONTAP on-premise.  Questo parametro è obbligatorio per i backend Amazon FSX per ONTAP e non può essere vuoto.	smb-share
nasType	<b>Deve essere impostato su smb.</b> se null, il valore predefinito è nfs.	smb
securityStyle	Stile di sicurezza per nuovi volumi.  <b>Deve essere impostato su ntfs oppure mixed Per volumi SMB.</b>	ntfs oppure mixed Per volumi SMB
unixPermissions	Per i nuovi volumi. <b>Deve essere lasciato vuoto per i volumi SMB.</b>	""

### Opzioni ed esempi di configurazione del NAS ONTAP

Scopri come creare e utilizzare i driver NAS ONTAP con la tua installazione Astra Trident. In questa sezione vengono forniti esempi di configurazione backend e dettagli per la mappatura dei 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	<p>Indirizzo IP di un cluster o LIF di gestione SVM</p> <p>È possibile specificare un nome di dominio completo (FQDN).</p> <p>Può essere impostato per utilizzare gli indirizzi IPv6 se Astra Trident è stato installato utilizzando il flag IPv6. Gli indirizzi IPv6 devono essere definiti tra parentesi quadre, ad esempio [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].</p> <p>Per uno switchover perfetto del MetroCluster, consulta il <a href="#">Esempio MetroCluster</a>.</p>	"10.0.0.1", "[2001:1234:abcd::fefe]"
dataLIF	<p>Indirizzo IP del protocollo LIF.</p> <p>Si consiglia di specificare dataLIF. Se non fornito, Astra Trident recupera i dati LIF dalla SVM. È possibile specificare un FQDN (Fully-qualified domain name) da utilizzare per le operazioni di montaggio NFS, consentendo di creare un DNS round-robin per il bilanciamento del carico tra più LIF di dati.</p> <p>Può essere modificato dopo l'impostazione iniziale. Fare riferimento a . .</p> <p>Può essere impostato per utilizzare gli indirizzi IPv6 se Astra Trident è stato installato utilizzando il flag IPv6. Gli indirizzi IPv6 devono essere definiti tra parentesi quadre, ad esempio [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].</p> <p><b>Ometti per MetroCluster.</b> vedere la <a href="#">Esempio MetroCluster</a>.</p>	Indirizzo specificato o derivato da SVM, se non specificato (non consigliato)
svm	<p>Macchina virtuale per lo storage da utilizzare</p> <p><b>Ometti per MetroCluster.</b> vedere la <a href="#">Esempio MetroCluster</a>.</p>	Derivato se un SVM managementLIF è specificato

Parametro	Descrizione	Predefinito
autoExportPolicy	Abilita la creazione e l'aggiornamento automatici dei criteri di esportazione [booleano].  Utilizzando il autoExportPolicy e. autoExportCIDRs Astra Trident è in grado di gestire automaticamente le policy di esportazione.	falso
autoExportCIDRs	Elenco di CIDR per filtrare gli IP dei nodi di Kubernetes rispetto a quando autoExportPolicy è attivato.  Utilizzando il autoExportPolicy e. autoExportCIDRs Astra Trident è in grado di gestire automaticamente le policy di esportazione.	["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	""
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	
storagePrefix	Prefisso utilizzato per il provisioning di nuovi volumi nella SVM. Non può essere aggiornato dopo l'impostazione	"trident"
limitAggregateUsage	Il provisioning non riesce se l'utilizzo è superiore a questa percentuale.  <b>Non si applica ad Amazon FSX per ONTAP</b>	"" (non applicato per impostazione predefinita)
limitVolumeSize	Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore.  Inoltre, limita le dimensioni massime dei volumi gestiti per qtree e LUN, oltre a qtreesPerFlexvol. Consente di personalizzare il numero massimo di qtree per FlexVol.	"" (non applicato per impostazione predefinita)
lunsPerFlexvol	LUN massimi per FlexVol, devono essere compresi nell'intervallo [50, 200]	"100"



Parametro	Descrizione	Predefinito
debugTraceFlags	<p>Flag di debug da utilizzare per la risoluzione dei problemi. Esempio, {"api":false, "method":true}</p> <p>Non utilizzare debugTraceFlags a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.</p>	nullo
nasType	<p>Configurare la creazione di volumi NFS o SMB.</p> <p>Le opzioni sono <code>nfs</code>, <code>smb</code> o nullo. L'impostazione su Null consente di impostare i volumi NFS come predefiniti.</p>	nfs
nfsMountOptions	<p>Elenco separato da virgole delle opzioni di montaggio NFS.</p> <p>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.</p> <p>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.</p>	""
qtreesPerFlexvol	<p>Qtree massimi per FlexVol, devono essere compresi nell'intervallo [50, 300]</p>	"200"
smbShare	<p>È possibile specificare una delle seguenti opzioni: Il nome di una condivisione SMB creata utilizzando la console di gestione Microsoft o l'interfaccia utente di ONTAP; un nome per consentire ad Astra Trident di creare la condivisione SMB; oppure è possibile lasciare vuoto il parametro per impedire l'accesso condiviso ai volumi.</p> <p>Questo parametro è facoltativo per ONTAP on-premise.</p> <p>Questo parametro è obbligatorio per i backend Amazon FSX per ONTAP e non può essere vuoto.</p>	smb-share

Parametro	Descrizione	Predefinito
useREST	<p>Parametro booleano per l'utilizzo delle API REST di ONTAP. <b>Anteprima tecnica</b></p> <p>useREST viene fornito come <b>anteprima tecnica</b> consigliata per ambienti di test e non per carichi di lavoro di produzione. Quando è impostato su <code>true</code>, Astra Trident utilizzerà le API REST di ONTAP per comunicare con il backend. Questa funzione richiede ONTAP 9.11.1 e versioni successive. Inoltre, il ruolo di accesso ONTAP utilizzato deve avere accesso a <code>ontap</code> applicazione. Ciò è soddisfatto dal predefinito <code>vsadmin</code> e <code>cluster-admin</code> ruoli.</p> <p>useREST Non è supportato con MetroCluster.</p>	falso

### Opzioni di configurazione back-end per il provisioning dei volumi

È possibile controllare il provisioning predefinito utilizzando queste opzioni in `defaults` della configurazione. Per un esempio, vedere gli esempi di configurazione riportati di seguito.

Parametro	Descrizione	Predefinito
<code>spaceAllocation</code>	Allocazione dello spazio per LUN	"vero"
<code>spaceReserve</code>	Modalità di prenotazione dello spazio; "nessuno" (sottile) o "volume" (spesso)	"nessuno"
<code>snapshotPolicy</code>	Policy di Snapshot da utilizzare	"nessuno"
<code>qosPolicy</code>	Gruppo di criteri QoS da assegnare per i volumi creati. Scegliere tra <code>qosPolicy</code> o <code>adaptiveQosPolicy</code> per pool di storage/backend	""
<code>adaptiveQosPolicy</code>	Gruppo di criteri QoS adattivi da assegnare per i volumi creati. Scegliere tra <code>qosPolicy</code> o <code>adaptiveQosPolicy</code> per pool di storage/backend.  Non supportato da <code>ontap-nas-Economy</code> .	""
<code>snapshotReserve</code>	Percentuale di volume riservato agli snapshot	"0" se <code>snapshotPolicy</code> è "nessuno", altrimenti ""
<code>splitOnClone</code>	Separare un clone dal suo padre al momento della creazione	"falso"

Parametro	Descrizione	Predefinito
encryption	<p>Abilitare NetApp Volume Encryption (NVE) sul nuovo volume; il valore predefinito è <code>false</code>. NVE deve essere concesso in licenza e abilitato sul cluster per utilizzare questa opzione.</p> <p>Se NAE è attivato sul backend, tutti i volumi forniti in Astra Trident saranno abilitati per NAE.</p> <p>Per ulteriori informazioni, fare riferimento a: "<a href="#">Come funziona Astra Trident con NVE e NAE</a>".</p>	"falso"
tieringPolicy	Criterio di tiering da utilizzare "nessuno"	"Solo Snapshot" per la configurazione SVM-DR pre-ONTAP 9,5
unixPermissions	Per i nuovi volumi	"777" per i volumi NFS; vuoto (non applicabile) per i volumi SMB
snapshotDir	Controlla l'accesso a <code>.snapshot directory</code>	"falso"
exportPolicy	Policy di esportazione da utilizzare	"predefinito"
securityStyle	<p>Stile di sicurezza per nuovi volumi.</p> <p>Supporto di NFS <code>mixed</code> e <code>unix</code> stili di sicurezza.</p> <p>Supporto SMB <code>mixed</code> e <code>ntfs</code> stili di sicurezza.</p>	<p>Il valore predefinito di NFS è <code>unix</code>.</p> <p>Il valore predefinito di SMB è <code>ntfs</code>.</p>



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.

## Esempi di provisioning di volumi

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  $2/0,5 = 10\text{GiB}$  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.

### Esempio di economia NAS ONTAP

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

### Esempio di FlexGroup NAS ONTAP

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

## Esempio MetroCluster

È possibile configurare il backend per evitare di dover aggiornare manualmente la definizione del backend dopo lo switchover e lo switchback durante ["Replica e recovery di SVM"](#).

Per uno switchover e uno switchback perfetto, specifica la SVM utilizzando `managementLIF` e omettere `dataLIF` e `svm` parametri. Ad esempio:

```
---  
version: 1  
storageDriverName: ontap-nas  
managementLIF: 192.168.1.66  
username: vsadmin  
password: password
```

## Esempio di volumi SMB

```
---  
version: 1  
backendName: ExampleBackend  
storageDriverName: ontap-nas  
managementLIF: 10.0.0.1  
nasType: smb  
securityStyle: ntfs  
unixPermissions: ""  
dataLIF: 10.0.0.2  
svm: svm_nfs  
username: vsadmin  
password: password
```

## Esempio di 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_
```

## Esempio di 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: password
nfsMountOptions: nfsvers=4
```

## Esempio di indirizzi IPv6

Questo esempio mostra managementLIF Utilizzando un indirizzo 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: password
```

## Esempio di Amazon FSX per ONTAP con volumi SMB

Il smbShare Il parametro è obbligatorio per FSX per ONTAP che utilizza volumi SMB.

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

## Esempi di backend con pool virtuali

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

Astra Trident imposta le etichette di provisioning nel campo "commenti". I commenti sono impostati su FlexVol per `ontap-nas` o FlexGroup per `ontap-nas-flexgroup`. Astra Trident copia tutte le etichette presenti su un pool virtuale nel volume di storage al momento del provisioning. Per comodità, gli amministratori dello storage possono definire le etichette per ogni pool virtuale e raggruppare i volumi per etichetta.



In questi esempi, alcuni dei pool di storage sono impostati in modo personalizzato `spaceReserve`, `spaceAllocation`, e `encryption` e alcuni pool sovrascrivono i valori predefiniti.

## Esempio di NAS ONTAP

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
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:
  department: legal
  creditpoints: '5000'
  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'
```

## Esempio di NAS FlexGroup ONTAP

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
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'
```

## Esempio di economia NAS ONTAP

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
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:
  protection: bronze
  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 fanno riferimento a [Esempi di backend con pool virtuali](#). 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 `protection-gold` StorageClass eseguirà il mapping al primo e al secondo pool virtuale in `ontap-nas-flexgroup` back-end. Questi sono gli unici pool che offrono una protezione di livello gold.

```
apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
  name: protection-gold  
provisioner: csi.trident.netapp.io  
parameters:  
  selector: "protection=gold"  
  fsType: "ext4"
```

- Il `protection-not-gold` StorageClass eseguirà il mapping al terzo e al quarto pool virtuale in `ontap-nas-flexgroup` back-end. Questi sono gli unici pool che offrono un livello di protezione diverso dall'oro.

```
apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
  name: protection-not-gold  
provisioner: csi.trident.netapp.io  
parameters:  
  selector: "protection!=gold"  
  fsType: "ext4"
```

- Il `app-mysqldb` StorageClass eseguirà il mapping al quarto pool virtuale in `ontap-nas` back-end. Questo è l'unico pool che offre la configurazione del pool di storage per l'applicazione di tipo `mysqldb`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- Il `protection-silver-creditpoints-20k` StorageClass eseguirà il mapping al terzo pool virtuale in `ontap-nas-flexgroup` back-end. Questo è l'unico pool che offre una protezione di livello Silver e 20000 punti di credito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- Il `creditpoints-5k` StorageClass eseguirà il mapping al terzo pool virtuale in `ontap-nas` il back-end e il secondo pool virtuale in `ontap-nas-economy` back-end. Queste sono le uniche offerte di pool con 5000 punti di credito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

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

#### **Aggiornare dataLIF dopo la configurazione iniziale**

È possibile modificare la LIF dei dati dopo la configurazione iniziale eseguendo il seguente comando per fornire al nuovo file JSON di back-end i dati aggiornati LIF.



```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-with-updated-dataLIF>
```



Se i PVC sono collegati a uno o più pod, è necessario abbassare tutti i pod corrispondenti e riportarli di nuovo in alto per rendere effettiva la nuova LIF dei dati.

## Amazon FSX per NetApp ONTAP

### 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 NetApp ONTAP. FSX per ONTAP 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 per ONTAP supporta le funzionalità del file system ONTAP e le API di amministrazione.

### Panoramica

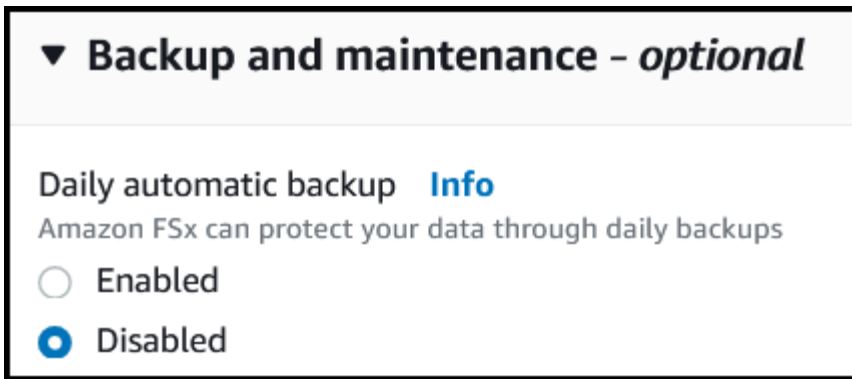
Un file system è la risorsa principale di Amazon FSX, simile a un cluster ONTAP on-premise. All'interno di ogni SVM è possibile creare uno o più volumi, ovvero contenitori di dati che memorizzano i file e le cartelle nel file system. Con Amazon FSX per NetApp ONTAP, Data ONTAP verrà fornito come file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

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

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.

### Considerazioni

- Volumi SMB:
  - I volumi SMB sono supportati utilizzando `ontap-nas` solo driver.
  - Astra Trident supporta volumi SMB montati su pod eseguiti solo su nodi Windows.
- I volumi creati sui file system Amazon FSX con backup automatici attivati non possono essere cancellati da Trident. Per eliminare i PVC, è necessario eliminare manualmente il volume FV e FSX per ONTAP. Per evitare questo problema:
  - Non utilizzare **creazione rapida** per creare il file system FSX per ONTAP. Il workflow di creazione rapida consente backup automatici e non offre un'opzione di opt-out.
  - Quando si utilizza **creazione standard**, disattivare il backup automatico. La disattivazione dei backup automatici consente a Trident di eliminare un volume senza ulteriori interventi manuali.



### Dettagli del driver FSX per ONTAP

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.

Per informazioni dettagliate sul driver, vedere ["Driver NAS"](#) e ["Driver SAN"](#).

### Autenticazione

Astra Trident offre due modalità di autenticazione.

- **Basato su certificato**: Astra Trident comunicherà con SVM sul file system FSX utilizzando un certificato installato sulla SVM.
- **Basato sulle credenziali**: È possibile utilizzare `fsxadmin` utente per il file system o l' `vsadmin` Configurato dall'utente per la SVM.



Astra Trident prevede di essere eseguito come `a. vsadmin` Utente SVM o come utente con un nome diverso che ha lo stesso ruolo. Amazon FSX per NetApp ONTAP ha un `fsxadmin` Utente che sostituisce in maniera limitata il `ONTAP admin` utente del cluster. Si consiglia vivamente di utilizzare `vsadmin` Con Astra Trident.

È possibile aggiornare i back-end per passare da un metodo basato su credenziali a un metodo basato su certificato. Tuttavia, se si tenta di fornire **credenziali e certificati**, la creazione del backend non riesce. Per passare a un metodo di autenticazione diverso, è necessario rimuovere il metodo esistente dalla configurazione di back-end.

Per ulteriori informazioni sull'attivazione dell'autenticazione, fare riferimento all'autenticazione per il tipo di driver in uso:

- ["Autenticazione NAS ONTAP"](#)

- ["Autenticazione SAN ONTAP"](#)

#### Trova ulteriori informazioni

- ["Documentazione di Amazon FSX per NetApp ONTAP"](#)
- ["Post del blog su Amazon FSX per NetApp ONTAP"](#)

### Integra Amazon FSX per NetApp ONTAP

Puoi integrare il file system Amazon FSX per NetApp ONTAP con Astra Trident per garantire che i cluster Kubernetes in esecuzione in Amazon Elastic Kubernetes Service (EKS) possano eseguire il provisioning di volumi persistenti di blocchi e file supportati da ONTAP.

#### Requisiti

Oltre a ["Requisiti di Astra Trident"](#) Per integrare FSX per ONTAP con Astra Trident, sono necessari:

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

- Astra Trident supporta volumi SMB montati su pod eseguiti solo su nodi Windows. Fare riferimento a [Preparatevi al provisioning dei volumi SMB](#) per ulteriori informazioni.

#### Integrazione dei driver ONTAP SAN e NAS



Se si configurano volumi SMB, è necessario leggere [Preparatevi al provisioning dei volumi SMB](#) prima di creare il backend.

#### Fasi

1. Implementare Astra Trident utilizzando uno dei ["metodi di implementazione"](#).
2. Raccogliere il nome DNS LIF di gestione 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 ["Autenticazione backend NAS"](#) oppure ["Autenticazione back-end SAN"](#).



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:

#### YAML

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

#### JSON

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "customBackendName",
  "managementLIF": "svm-XXXXXXXXXXXXXXXXXXXX.fs-
XXXXXXXXXXXXXXXXXXXX.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

#### Preparatevi al provisioning dei volumi SMB

È possibile eseguire il provisioning dei volumi SMB utilizzando `ontap-nas` driver. Prima di completare [Integrazione dei driver ONTAP SAN e NAS](#) completare i seguenti passaggi.

#### Prima di iniziare

Prima di eseguire il provisioning di volumi SMB utilizzando `ontap-nas` driver, è necessario disporre di quanto segue.

- Un cluster Kubernetes con un nodo controller Linux e almeno un nodo di lavoro Windows che esegue

Windows Server 2019. Astra Trident supporta volumi SMB montati su pod eseguiti solo su nodi Windows.

- Almeno un segreto Astra Trident contenente le credenziali Active Directory. Per generare un segreto smbcreds:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Proxy CSI configurato come servizio Windows. Per configurare un `csi-proxy`, fare riferimento a ["GitHub: Proxy CSI"](#) oppure ["GitHub: Proxy CSI per Windows"](#) Per i nodi Kubernetes in esecuzione su Windows.

## Fasi

1. Creare condivisioni SMB. È possibile creare le condivisioni amministrative SMB in due modi utilizzando ["Console di gestione Microsoft"](#) Snap-in cartelle condivise o utilizzo dell'interfaccia CLI di ONTAP. Per creare le condivisioni SMB utilizzando la CLI ONTAP:

- a. Se necessario, creare la struttura del percorso di directory per la condivisione.

Il `vserver cifs share create` il comando controlla il percorso specificato nell'opzione `-path` durante la creazione della condivisione. Se il percorso specificato non esiste, il comando non riesce.

- b. Creare una condivisione SMB associata alla SVM specificata:

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

- c. Verificare che la condivisione sia stata creata:

```
vserver cifs share show -share-name share_name
```



Fare riferimento a ["Creare una condivisione SMB"](#) per informazioni dettagliate.

2. Quando si crea il backend, è necessario configurare quanto segue per specificare i volumi SMB. Per tutte le opzioni di configurazione backend FSX per ONTAP, fare riferimento a ["FSX per le opzioni di configurazione e gli esempi di ONTAP"](#).

Parametro	Descrizione	Esempio
smbShare	È possibile specificare una delle seguenti opzioni: Il nome di una condivisione SMB creata utilizzando la console di gestione Microsoft o l'interfaccia utente di ONTAP o un nome per consentire ad Astra Trident di creare la condivisione SMB.  Questo parametro è obbligatorio per i backend Amazon FSX per ONTAP.	smb-share
nasType	<b>Deve essere impostato su smb.</b> se null, il valore predefinito è nfs.	smb
securityStyle	Stile di sicurezza per nuovi volumi.  <b>Deve essere impostato su ntfs oppure mixed Per volumi SMB.</b>	ntfs oppure mixed Per volumi SMB
unixPermissions	Per i nuovi volumi. <b>Deve essere lasciato vuoto per i volumi SMB.</b>	""

## FSX per le opzioni di configurazione e gli esempi di ONTAP

Scopri le opzioni di configurazione back-end per Amazon FSX per ONTAP. Questa sezione fornisce esempi di configurazione back-end.

### Opzioni di configurazione back-end

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

Parametro	Descrizione	Esempio
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

Parametro	Descrizione	Esempio
managementLIF	<p>Indirizzo IP di un cluster o LIF di gestione SVM</p> <p>È possibile specificare un nome di dominio completo (FQDN).</p> <p>Può essere impostato per utilizzare gli indirizzi IPv6 se Astra Trident è stato installato utilizzando il flag IPv6. Gli indirizzi IPv6 devono essere definiti tra parentesi quadre, ad esempio [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].</p>	"10.0.0.1", "[2001:1234:abcd::fefe]"
dataLIF	<p>Indirizzo IP del protocollo LIF.</p> <p><b>Driver NAS ONTAP:</b> Si consiglia di specificare dataLIF. Se non fornito, Astra Trident recupera i dati LIF dalla SVM. È possibile specificare un FQDN (Fully-qualified domain name) da utilizzare per le operazioni di montaggio NFS, consentendo di creare un DNS round-robin per il bilanciamento del carico tra più LIF di dati. Può essere modificato dopo l'impostazione iniziale. Fare riferimento a . .</p> <p><b>Driver SAN ONTAP:</b> Non specificare iSCSI. Astra Trident utilizza la mappa LUN selettiva di ONTAP per rilevare le LIF iSCSI necessarie per stabilire una sessione multi-percorso. Viene generato un avviso se dataLIF è esplicitamente definito.</p> <p>Può essere impostato per utilizzare gli indirizzi IPv6 se Astra Trident è stato installato utilizzando il flag IPv6. Gli indirizzi IPv6 devono essere definiti tra parentesi quadre, ad esempio [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].</p>	

Parametro	Descrizione	Esempio
autoExportPolicy	<p>Abilita la creazione e l'aggiornamento automatici dei criteri di esportazione [booleano].</p> <p>Utilizzando il autoExportPolicy e. autoExportCIDRs Astra Trident è in grado di gestire automaticamente le policy di esportazione.</p>	false
autoExportCIDRs	<p>Elenco di CIDR per filtrare gli IP dei nodi di Kubernetes rispetto a quando autoExportPolicy è attivato.</p> <p>Utilizzando il autoExportPolicy e. autoExportCIDRs Astra Trident è in grado di gestire automaticamente le policy di esportazione.</p>	"["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	""
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 o alla SVM. Utilizzato per l'autenticazione basata su credenziali. Ad esempio, vsadmin.	
password	Password per la connessione al cluster o alla SVM. Utilizzato per l'autenticazione basata su credenziali.	
svm	Macchina virtuale per lo storage da utilizzare	Derivato se viene specificato un LIF di gestione SVM.



Parametro	Descrizione	Esempio
storagePrefix	<p>Prefisso utilizzato per il provisioning di nuovi volumi nella SVM.</p> <p>Impossibile modificare dopo la creazione. Per aggiornare questo parametro, è necessario creare un nuovo backend.</p>	trident
limitAggregateUsage	<p><b>Non specificare Amazon FSX per NetApp ONTAP.</b></p> <p>Il fornito fsxadmin e. vsadmin Non includere le autorizzazioni necessarie per recuperare l'utilizzo aggregato e limitarlo utilizzando Astra Trident.</p>	Non utilizzare.
limitVolumeSize	<p>Fallire il provisioning se la dimensione del volume richiesta è superiore a questo valore.</p> <p>Inoltre, limita le dimensioni massime dei volumi gestiti per qtree e LUN, oltre a qtreesPerFlexvol Consente di personalizzare il numero massimo di qtree per FlexVol.</p>	"" (non applicato per impostazione predefinita)
lunsPerFlexvol	<p>Il numero massimo di LUN per FlexVol deve essere compreso nell'intervallo [50, 200].</p> <p>Solo SAN.</p>	100
debugTraceFlags	<p>Flag di debug da utilizzare per la risoluzione dei problemi. Ad esempio, {"api":false,} method":true</p> <p>Non utilizzare debugTraceFlags a meno che non si stia eseguendo la risoluzione dei problemi e non si richieda un dump dettagliato del log.</p>	nullo

Parametro	Descrizione	Esempio
nfsMountOptions	<p>Elenco separato da virgole delle opzioni di montaggio NFS.</p> <p>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.</p> <p>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.</p>	""
nasType	<p>Configurare la creazione di volumi NFS o SMB.</p> <p>Le opzioni sono <code>nfs</code>, <code>smb`</code> o <code>nullo</code>.</p> <p><b>Deve essere impostato su `smb</b> Per i volumi SMB. l'impostazione su Null imposta come predefinita i volumi NFS.</p>	nfs
qtreesPerFlexvol	Qtree massimi per FlexVol, devono essere compresi nell'intervallo [50, 300]	200
smbShare	<p>È possibile specificare una delle seguenti opzioni: Il nome di una condivisione SMB creata utilizzando la console di gestione Microsoft o l'interfaccia utente di ONTAP o un nome per consentire ad Astra Trident di creare la condivisione SMB.</p> <p>Questo parametro è obbligatorio per i backend Amazon FSX per ONTAP.</p>	smb-share

Parametro	Descrizione	Esempio
useREST	<p>Parametro booleano per l'utilizzo delle API REST di ONTAP.</p> <p><b>Anteprima tecnica</b></p> <p>useREST viene fornito come <b>anteprima tecnica</b> consigliata per ambienti di test e non per carichi di lavoro di produzione. Quando è impostato su <code>true</code>, Astra Trident utilizzerà le API REST di ONTAP per comunicare con il backend.</p> <p>Questa funzione richiede ONTAP 9.11.1 e versioni successive. Inoltre, il ruolo di accesso ONTAP utilizzato deve avere accesso a <code>ontap</code> applicazione. Ciò è soddisfatto dal predefinito <code>vsadmin</code> e <code>cluster-admin</code> ruoli.</p>	false

### Aggiornare dataLIF dopo la configurazione iniziale

È possibile modificare la LIF dei dati dopo la configurazione iniziale eseguendo il seguente comando per fornire al nuovo file JSON di back-end i dati aggiornati LIF.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-with-updated-dataLIF>
```



Se i PVC sono collegati a uno o più pod, è necessario abbassare tutti i pod corrispondenti e riportarli di nuovo in alto per rendere effettiva la nuova LIF dei dati.

### Opzioni di configurazione back-end per il provisioning dei volumi

È possibile controllare il provisioning predefinito utilizzando queste opzioni in `defaults` della configurazione. Per un esempio, vedere gli esempi di configurazione riportati di seguito.

Parametro	Descrizione	Predefinito
spaceAllocation	Allocazione dello spazio per LUN	true
spaceReserve	Modalità di riserva dello spazio; "nessuno" (sottile) o "volume" (spesso)	none
snapshotPolicy	Policy di Snapshot da utilizzare	none

Parametro	Descrizione	Predefinito
qosPolicy	<p>Gruppo di criteri QoS da assegnare per i volumi creati. Scegliere una delle opzioni qosPolicy o adaptiveQosPolicy per pool di storage o backend.</p> <p>L'utilizzo di gruppi di policy QoS con Astra Trident richiede ONTAP 9.8 o versione successiva.</p> <p>Si consiglia di utilizzare un gruppo di policy QoS non condiviso e di assicurarsi che il gruppo di policy venga 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.</p>	""
adaptiveQosPolicy	<p>Gruppo di criteri QoS adattivi da assegnare per i volumi creati. Scegliere una delle opzioni qosPolicy o adaptiveQosPolicy per pool di storage o backend.</p> <p>Non supportato da ontap-nas-Economy.</p>	""
snapshotReserve	Percentuale di volume riservato agli snapshot "0"	Se snapshotPolicy è none, else ""
splitOnClone	Separare un clone dal suo padre al momento della creazione	false
encryption	<p>Abilitare NetApp Volume Encryption (NVE) sul nuovo volume; il valore predefinito è false. NVE deve essere concesso in licenza e abilitato sul cluster per utilizzare questa opzione.</p> <p>Se NAE è attivato sul backend, tutti i volumi forniti in Astra Trident saranno abilitati per NAE.</p> <p>Per ulteriori informazioni, fare riferimento a: "<a href="#">Come funziona Astra Trident con NVE e NAE</a>".</p>	false

Parametro	Descrizione	Predefinito
luksEncryption	Attivare la crittografia LUKS. Fare riferimento a. <a href="#">"Utilizzo di Linux Unified Key Setup (LUKS)"</a> .  Solo SAN.	""
tieringPolicy	Policy di tiering da utilizzare none	snapshot-only Per la configurazione SVM-DR precedente a ONTAP 9.5
unixPermissions	Per i nuovi volumi.  <b>Lasciare vuoto per i volumi SMB.</b>	""
securityStyle	Stile di sicurezza per nuovi volumi.  Supporto di NFS mixed e. unix stili di sicurezza.  Supporto SMB mixed e. ntfs stili di sicurezza.	Il valore predefinito di NFS è unix.  Il valore predefinito di SMB è ntfs.

### Esempio

Utilizzo di nasType, node-stage-secret-name, e. node-stage-secret-namespace, È possibile specificare un volume SMB e fornire le credenziali Active Directory richieste. I volumi SMB sono supportati utilizzando ontap-nas solo driver.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: nas-smb-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

### Configurare il componente aggiuntivo Astra Trident EKS versione 23,10 sul cluster EKS

Astra Trident ottimizza la gestione dello storage di Amazon FSX per NetApp ONTAP in Kubernetes per permettere a sviluppatori e amministratori di concentrarsi sull'implementazione dell'applicazione. Il componente aggiuntivo Astra Trident EKS include le più recenti patch di sicurezza, correzioni di bug ed è convalidato da AWS per funzionare con Amazon EKS. Il componente aggiuntivo EKS ti consente di garantire in modo coerente che i tuoi cluster Amazon EKS siano sicuri e stabili e di ridurre la quantità di lavoro da svolgere per installare, configurare e aggiornare i componenti aggiuntivi.

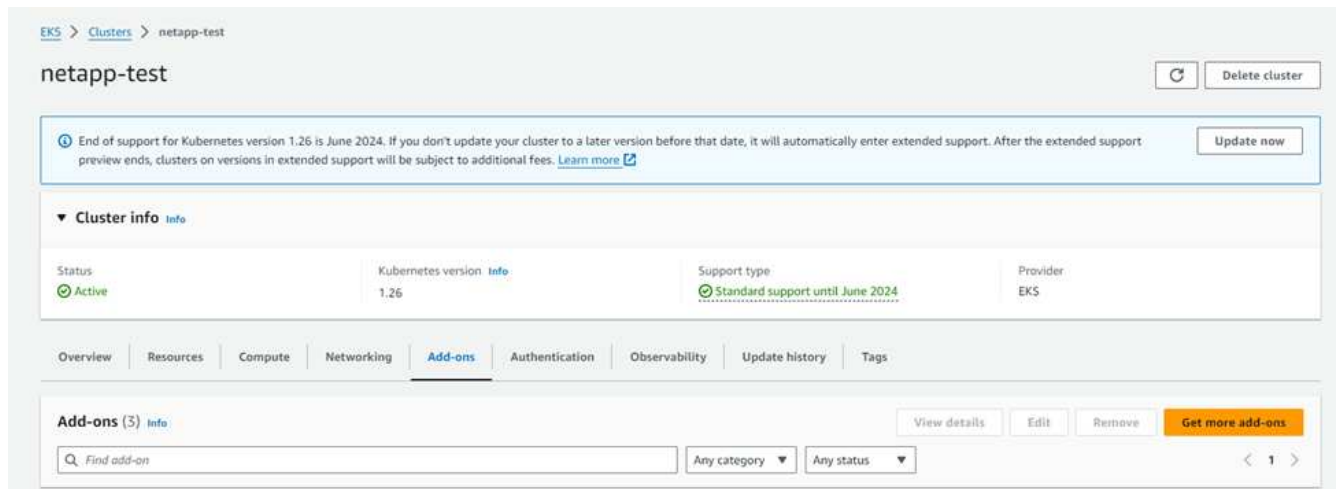
## Prerequisiti

Prima di configurare il componente aggiuntivo Astra Trident per AWS EKS, assicurati di disporre di quanto segue:

- Un account cluster Amazon EKS con abbonamento add-on
- Autorizzazioni AWS nel marketplace AWS:  
"aws-marketplace:ViewSubscriptions",  
"aws-marketplace:Subscribe",  
"aws-marketplace:Unsubscribe"
- Tipo di ami: Amazon Linux 2 (AL2\_x86\_64) o Amazon Linux 2 Arm (AL2\_ARM\_64)
- Tipo di nodo: AMD o ARM
- Un file system Amazon FSX per NetApp ONTAP esistente

## Fasi

1. Sul tuo cluster EKS Kubernetes, accedi alla scheda **Add-on**.

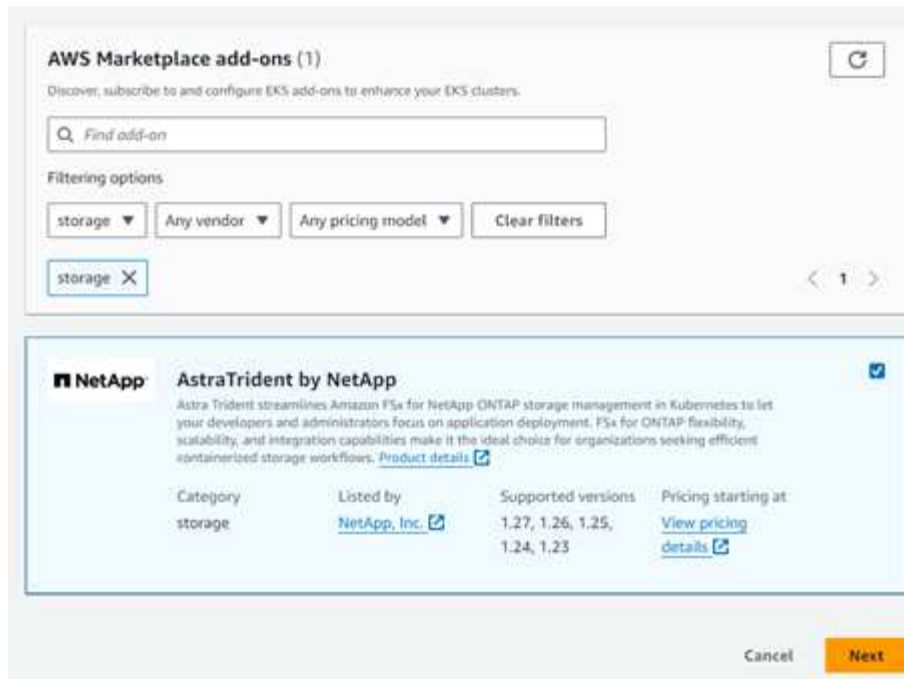


The screenshot shows the AWS Management Console interface for an EKS cluster named 'netapp-test'. The breadcrumb navigation is 'EKS > Clusters > netapp-test'. The cluster name 'netapp-test' is displayed at the top left, with a refresh icon and a 'Delete cluster' button on the right. A blue notification banner at the top states: 'End of support for Kubernetes version 1.26 is June 2024. If you don't update your cluster to a later version before that date, it will automatically enter extended support. After the extended support preview ends, clusters on versions in extended support will be subject to additional fees. Learn more'. An 'Update now' button is located on the right of this banner. Below the banner is the 'Cluster info' section, which includes a table with the following data:

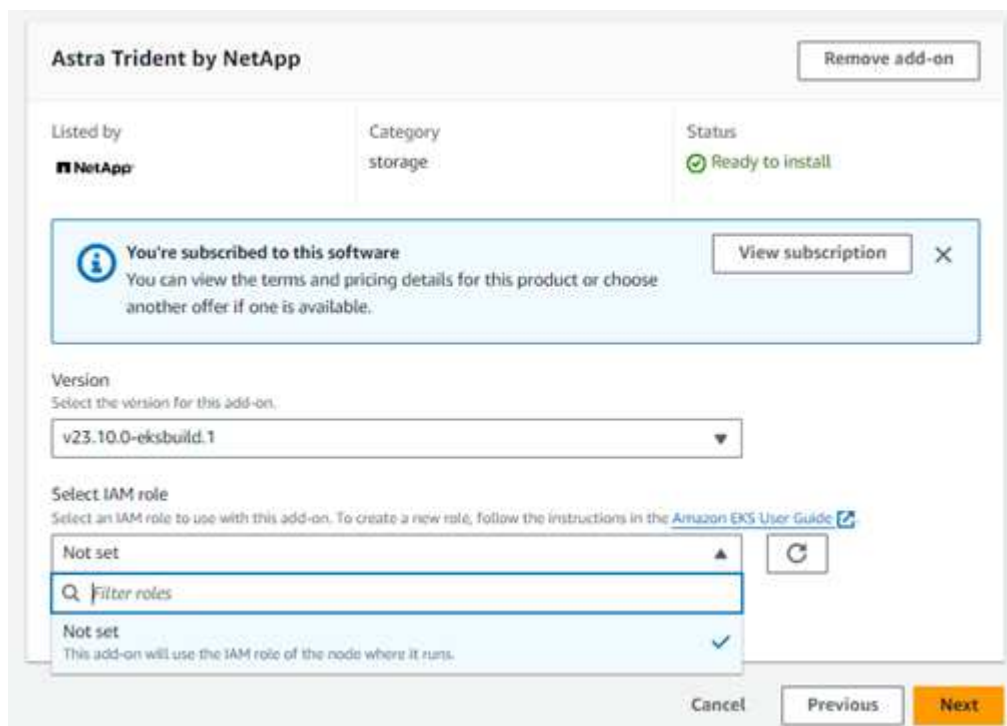
Cluster info			
Status: Active	Kubernetes version: 1.26	Support type: Standard support until June 2024	Provider: EKS

Below the table is a navigation bar with tabs: Overview, Resources, Compute, Networking, Add-ons (selected), Authentication, Observability, Update history, and Tags. The 'Add-ons' section is active, showing 'Add-ons (3) Info'. It includes a search bar with the placeholder 'Find add-on', two dropdown menus for 'Any category' and 'Any status', and buttons for 'View details', 'Edit', 'Remove', and 'Get more add-ons'.

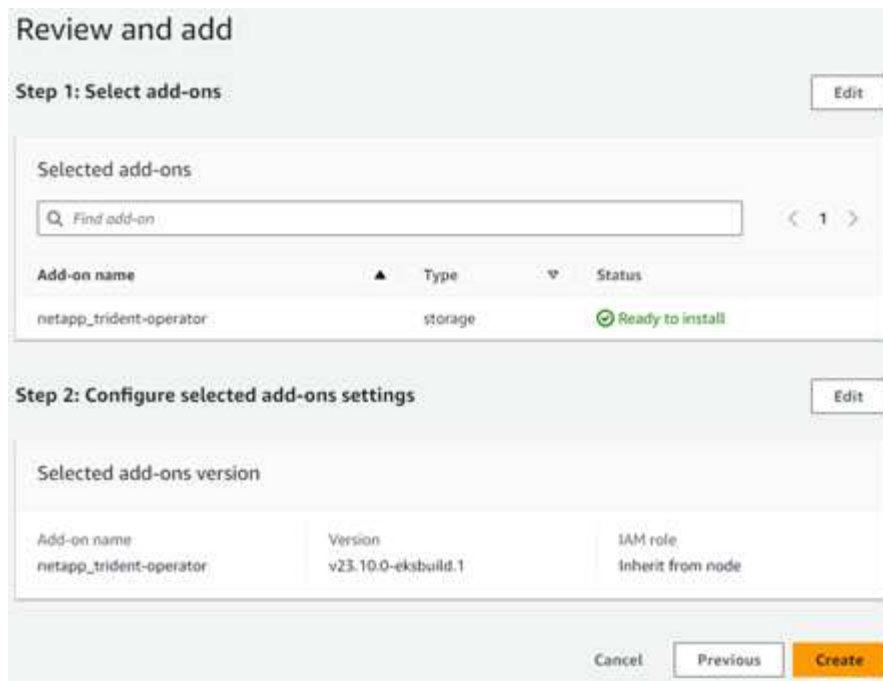
2. Vai su **componenti aggiuntivi di AWS Marketplace** e scegli la categoria *storage*.



3. Individua **AstraTrident by NetApp** e seleziona la casella di controllo per il componente aggiuntivo Astra Trident.
4. Scegliere la versione desiderata del componente aggiuntivo.



5. Selezionare l'opzione ruolo IAM per ereditare dal nodo.
6. Configurare eventuali impostazioni opzionali secondo necessità e selezionare **Avanti**.



7. Selezionare **Crea**.
8. Verificare che lo stato del componente aggiuntivo sia *attivo*.



**Installare/disinstallare il componente aggiuntivo Astra Trident EKS utilizzando la CLI**

**Installare il componente aggiuntivo Astra Trident EKS utilizzando la CLI:**

I seguenti comandi di esempio installano il componente aggiuntivo Astra Trident EKS:

```
eksctl create addon --cluster K8s-arm --name netapp_trident-operator --version v23.10.0-eksbuild.1
eksctl create addon --cluster K8s-arm --name netapp_trident-operator --version v23.10.0-eksbuild.1 (con una versione dedicata)
```

**Disinstallare il componente aggiuntivo Astra Trident EKS utilizzando la CLI:**

Il seguente comando disinstalla il componente aggiuntivo Astra Trident EKS:

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

## 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 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
ONTAP	ClientPrivateKey	Valore codificato in base64 della chiave privata del client. Utilizzato per l'autenticazione basata su certificato

Descrizione dei campi segreti della piattaforma di storage	Segreto	Descrizione dei campi
ONTAP	ChapNomeUtente	Nome utente inbound. Obbligatorio se useCHAP=true. Per ontap-san e. ontap-san-economy
ONTAP	ChapInitialiatorSecret	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	ChapTargetInitialiatorSecret	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 Uid di CR.
- **Unbound:** Rappresentato utilizzando "". Il TridentBackendConfig l'oggetto non è associato a un backend. Tutti i recenti 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](#)".

## Gestire i backend

### 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).
  - È possibile eseguire l'aggiornamento `TridentBackendConfig` Oggetti direttamente tramite Kubernetes usando il seguente comando:

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

- In alternativa, è possibile apportare modifiche all'esistente `TridentBackendConfig` CR utilizzando 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.

### Opzioni per la gestione dei backend

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 hanno un `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 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 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.

# Creare e gestire classi di archiviazione

## Creare una classe di storage

Configurare un oggetto Kubernetes `StorageClass` e creare una classe storage per istruire Astra Trident su come eseguire il provisioning dei volumi.

### Configurare un oggetto Kubernetes `StorageClass`

Il "[Oggetto Kubernetes StorageClass](#)" Identifica Astra Trident come provisioner utilizzato per quella classe istruisce Astra Trident su come eseguire il provisioning di un volume. Ad esempio:



```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters:
  <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Fare riferimento a ["Kubernetes e Trident Objects"](#) per informazioni sulle modalità di interazione delle classi di storage con `PersistentVolumeClaim` E parametri per controllare come Astra Trident esegue il provisioning dei volumi.

### Creare una classe di storage

Dopo aver creato l'oggetto `StorageClass`, è possibile creare la classe storage. [Campioni di classe di conservazione](#) fornisce alcuni esempi di base che è possibile utilizzare o modificare.

#### Fasi

1. Si tratta di un oggetto Kubernetes, lo utilizza `kubectl` Per crearlo in Kubernetes.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

2. Ora dovrebbe essere visualizzata una classe di storage **Basic-csi** in Kubernetes e Astra Trident, mentre Astra Trident avrebbe scoperto i pool sul backend.

```

kubect1 get sc basic-csi
NAME          PROVISIONER          AGE
basic-csi     csi.trident.netapp.io 15h

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}

```

### Campioni di classe di conservazione

Astra Trident offre ["definizioni semplici delle classi di archiviazione per backend specifici"](#).

In alternativa, è possibile modificare `sample-input/storage-class-csi.yaml.template` file fornito con il programma di installazione e sostituirlo `BACKEND_TYPE` con il nome del driver di storage.

```

./tridentctl -n trident get backend
+-----+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |           UUID           |
STATE | VOLUMES |
+-----+-----+-----+-----+
+-----+-----+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |         0 |
+-----+-----+-----+-----+
+-----+-----+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml

```

## Gestire le classi di storage

È possibile visualizzare le classi di storage esistenti, impostare una classe di storage predefinita, identificare il backend della classe di storage ed eliminare le classi di storage.

### 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 deve essere presente una sola classe di archiviazione predefinita alla 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}]'
```

## 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 le `StorageClasses` esistenti e ricrearle con `parameters.fsType` specificato.

## Provisioning e gestione dei volumi

### Provisioning di un volume

Creare un `PersistentVolume` (PV) e un `PersistentVolumeClaim` (PVC) che utilizza `Kubernetes StorageClass` configurato per richiedere l'accesso al PV. È quindi possibile montare il PV su un pod.

#### Panoramica

R "*PersistentVolume*" (PV) è una risorsa di storage fisico fornita dall'amministratore del cluster su un cluster Kubernetes. Il "*PersistentVolumeClaim*" (PVC) è una richiesta di accesso a `PersistentVolume` sul cluster.

Il PVC può essere configurato per richiedere la memorizzazione di una determinata dimensione o modalità di accesso. Utilizzando `StorageClass` associato, l'amministratore del cluster può controllare più delle dimensioni di `PersistentVolume` e della modalità di accesso, ad esempio le prestazioni o il livello di servizio.

Dopo aver creato PV e PVC, è possibile montare il volume in un pod.

#### Manifesti campione

#### Manifesto di esempio di `PersistentVolume`

Questo manifesto di esempio mostra un PV di base di 10Gi associato a `StorageClass basic-csi`.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-storage
  labels:
    type: local
spec:
  storageClassName: basic-csi
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/my/host/path"
```

## Manifesti di campioni PersistentVolumeClaim

Questi esempi mostrano le opzioni di configurazione di base del PVC.

### PVC con accesso RWO

Questo esempio mostra un PVC di base con accesso RWO associato a un nome StorageClass `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

### PVC con NVMe/TCP

Questo esempio mostra un PVC di base per NVMe/TCP con accesso RWO associato a una StorageClass denominata `protection-gold`.

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san-nvme
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 300Mi
  storageClassName: protection-gold
```

## Campioni manifesti pod

Questi esempi mostrano le configurazioni di base per collegare il PVC a un pod.

### Configurazione di base

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
        claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

## Configurazione NVMe/TCP di base

```
---
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
    name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    resources: {}
    volumeMounts:
    - mountPath: "/usr/share/nginx/html"
      name: task-pv-storage
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  volumes:
  - name: task-pv-storage
    persistentVolumeClaim:
      claimName: pvc-san-nvme
```

## Creare PV e PVC

### Fasi

1. Creare il PV.

```
kubectl create -f pv.yaml
```

2. Verificare lo stato PV.

```
kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM
STORAGECLASS  REASON    AGE
pv-storage    4Gi       RWO           Retain          Available
7s
```

3. Creare il PVC.



```
kubectl create -f pvc.yaml
```

#### 4. Verificare lo stato del PVC.

```
kubectl get pvc
NAME          STATUS VOLUME          CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage  Bound  pv-name 2Gi          RWO          5m
```

#### 5. Montare il volume in un pod.

```
kubectl create -f pv-pod.yaml
```



È possibile monitorare l'avanzamento utilizzando `kubectl get pod --watch`.

#### 6. Verificare che il volume sia montato su `/my/mount/path`.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

#### 7. A questo punto è possibile eliminare il pod. L'applicazione Pod non esisterà più, ma il volume rimarrà.

```
kubectl delete pod task-pv-pod
```

Fare riferimento a ["Kubernetes e Trident Objects"](#) per informazioni sulle modalità di interazione delle classi di storage con `PersistentVolumeClaim` E parametri per controllare come Astra Trident esegue il provisioning dei volumi.

## 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.

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

Modificare la definizione StorageClass per impostare `allowVolumeExpansion` campo a `true`.

```
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

Modificare la definizione PVC e aggiornare `spec.resources.requests.storage` per riflettere le nuove dimensioni desiderate, che devono essere superiori alle dimensioni originali.

```
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).

```

kubect1 get pvc
NAME          STATUS      VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound       pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi
RWO          ontap-san    8s

kubect1 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

Collegare il PV a un pod affinché venga 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.

In questo esempio, viene creato un pod che utilizza `san-pvc`.

```
kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
ubuntu-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:    ubuntu-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:

```

kubect1 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
kubect1 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, 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 `kubect1 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 1GiB:

```
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:



```

kubect1 get pvc ontapnas20mb
NAME                STATUS      VOLUME
CAPACITY  ACCESS MODES  STORAGECLASS  AGE
ontapnas20mb  Bound        pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  1Gi
RWO                ontapnas                4m44s

kubect1 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`.

### Panoramica e considerazioni

È possibile importare un volume in Astra Trident per:

- Containerizzare un'applicazione e riutilizzare il set di dati esistente
- Utilizzare un clone di un set di dati per un'applicazione temporanea
- Ricostruire un cluster Kubernetes guasto
- Migrazione dei dati delle applicazioni durante il disaster recovery

### Considerazioni

Prima di importare un volume, esaminare le seguenti considerazioni.

- Astra Trident può importare solo volumi ONTAP di tipo RW (Read-write). I volumi di tipo DP (data Protection) sono volumi di destinazione SnapMirror. Prima di importare il volume in Astra Trident, è necessario interrompere la relazione di mirroring.

- Si consiglia di importare volumi senza connessioni attive. Per importare un volume utilizzato attivamente, clonare il volume ed eseguire l'importazione.



Ciò è particolarmente importante per i volumi a blocchi, in quanto Kubernetes non sarebbe a conoscenza della connessione precedente e potrebbe facilmente collegare un volume attivo a un pod. Ciò può causare il danneggiamento dei dati.

- Tuttavia `StorageClass` Deve essere specificato su PVC, Astra Trident non utilizza questo parametro durante l'importazione. Le classi di storage vengono utilizzate durante la creazione del volume per selezionare i pool disponibili in base alle caratteristiche dello storage. Poiché il volume esiste già, durante l'importazione non è richiesta alcuna selezione del pool. Pertanto, l'importazione non avrà esito negativo anche se il volume esiste in un backend o in un pool che non corrisponde alla classe di storage specificata nel PVC.
- La dimensione del volume esistente viene determinata e impostata 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.
- Per impostazione predefinita, Astra Trident gestisce il PVC e rinomina il FlexVol e il LUN sul backend. È possibile superare il `--no-manage` contrassegna per importare un volume non gestito. Se si utilizza `--no-manage`, Astra Trident non esegue operazioni aggiuntive sul PVC o sul PV per il ciclo di vita degli oggetti. Il volume di storage non viene cancellato quando il PV viene cancellato e 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.

## Importare un volume

È possibile utilizzare `tridentctl import` per importare un volume.

### Fasi

1. Creare il file PVC (Persistent Volume Claim) (ad esempio, `pvc.yaml`) Che verrà utilizzato per creare il PVC. Il file PVC deve includere `name`, `namespace`, `accessModes`, e `storageClassName`. In alternativa, è possibile specificare `unixPermissions` Nella definizione di PVC.

Di seguito viene riportato un esempio di specifica minima:

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



Non includere parametri aggiuntivi come il nome PV o le dimensioni del volume. Questo può causare l'errore del comando di importazione.

2. Utilizzare `tridentctl import volume` Per specificare il nome del backend Astra Trident contenente il volume e il nome che identifica in modo univoco il volume nello storage (ad esempio: ONTAP FlexVol, volume elemento, percorso Cloud Volumes Service). Il `-f` Argomento necessario per specificare il percorso del file PVC.

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

## Esempi

Consultare i seguenti esempi di importazione di volumi per i driver supportati.

### NAS ONTAP e NAS FlexGroup ONTAP

Astra Trident supporta l'importazione di volumi utilizzando `ontap-nas` e `ontap-nas-flexgroup` driver.



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

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.

### Esempi di NAS ONTAP

Di seguito viene illustrato un esempio di importazione di un volume gestito e di un volume non gestito.

## Volume gestito

Nell'esempio seguente viene importato un volume denominato `managed_volume` su un backend denominato `ontap_nas`:

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

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

## Volume non gestito

Quando si utilizza `--no-manage` Argomento: Astra Trident non rinomina il volume.

L'esempio seguente importa `unmanaged_volume` su `ontap_nas` back-end:

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

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

## ONTAP SAN

Astra Trident supporta l'importazione di volumi utilizzando `ontap-san` driver. L'importazione di un volume non è supportata con `ontap-san-economy` driver.

Astra Trident può 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. Astra Trident importa il FlexVol e lo associa alla definizione del PVC.

## Esempi DI SAN ONTAP

Di seguito viene illustrato un esempio di importazione di un volume gestito e di un volume non gestito.

### Volume gestito

Per i volumi gestiti, Astra Trident rinomina FlexVol in `pvc-<uuid>` E il LUN all'interno di FlexVol a `lun0`.

Nell'esempio riportato di seguito viene importato il `ontap-san-managed` FlexVol presente su `ontap_san_default` back-end:

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

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

### Volume non gestito

L'esempio seguente importa `unmanaged_example_volume` su `ontap_san` back-end:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
block	pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228	e3275890-7d80-4af6-90cc-c7a0759f555a	1.0 GiB	online	san-blog	false

Se si dispone DI LUN mappati a igroups che condividono un IQN con un nodo Kubernetes IQN, come mostrato nell'esempio seguente, viene visualizzato l'errore: `LUN already mapped to initiator(s) in this group`. Per importare il volume, è necessario rimuovere l'iniziatore o annullare la mappatura del LUN.

Vserver	Igroup	Protocol	OS Type	Initiators
svm0	k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3	iscsi	linux	iqn.1994-05.com.redhat:4c2e1cf35e0
svm0	unmanaged-example-igroup	mixed	linux	iqn.1994-05.com.redhat:4c2e1cf35e0

### Elemento

Astra Trident supporta il software NetApp Element e l'importazione di volumi NetApp HCI utilizzando `solidfire-san` driver.



Il driver Element supporta nomi di volumi duplicati. Tuttavia, Astra Trident restituisce un errore se sono presenti nomi di volumi duplicati. Come soluzione alternativa, clonare il volume, fornire un nome di volume univoco e importare il volume clonato.

### Esempio di elemento

Nell'esempio seguente viene importato un `element-managed` volume sul back-end `element_default`.

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

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

### Piattaforma Google Cloud

Astra Trident supporta l'importazione di volumi utilizzando `gcp-cvs` driver.



Per importare un volume supportato da NetApp Cloud Volumes Service in Google Cloud Platform, identificare il volume in base al relativo percorso. 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`.

### Esempio di piattaforma Google Cloud

Nell'esempio seguente viene importato un `gcp-cvs` volume sul back-end `gcpcvs_YEppr` con il percorso del volume di `adroit-jolly-swift`.

```
tridentctl import volume gcpcvs_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 | gcp-storage   | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true         |
+-----+-----+-----+
+-----+-----+-----+-----+
```

### Azure NetApp Files

Astra Trident supporta l'importazione di volumi utilizzando `azure-netapp-files` driver.



Per importare un volume Azure NetApp Files, identificare il volume in base al relativo percorso. Il percorso del volume è la parte del percorso di esportazione del volume dopo `:/`. Ad esempio, se il percorso di montaggio è `10.0.0.2:/importvoll1`, il percorso del volume è `importvoll1`.

### Esempio di Azure NetApp Files

Nell'esempio seguente viene importato un `azure-netapp-files` volume sul back-end `azurenetappfiles_40517` con il percorso del volume `importvoll1`.

```
tridentctl import volume azurenetappfiles_40517 importvoll1 -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         |
+-----+-----+-----+
+-----+-----+-----+-----+
```

## Condividere un volume NFS tra spazi dei nomi

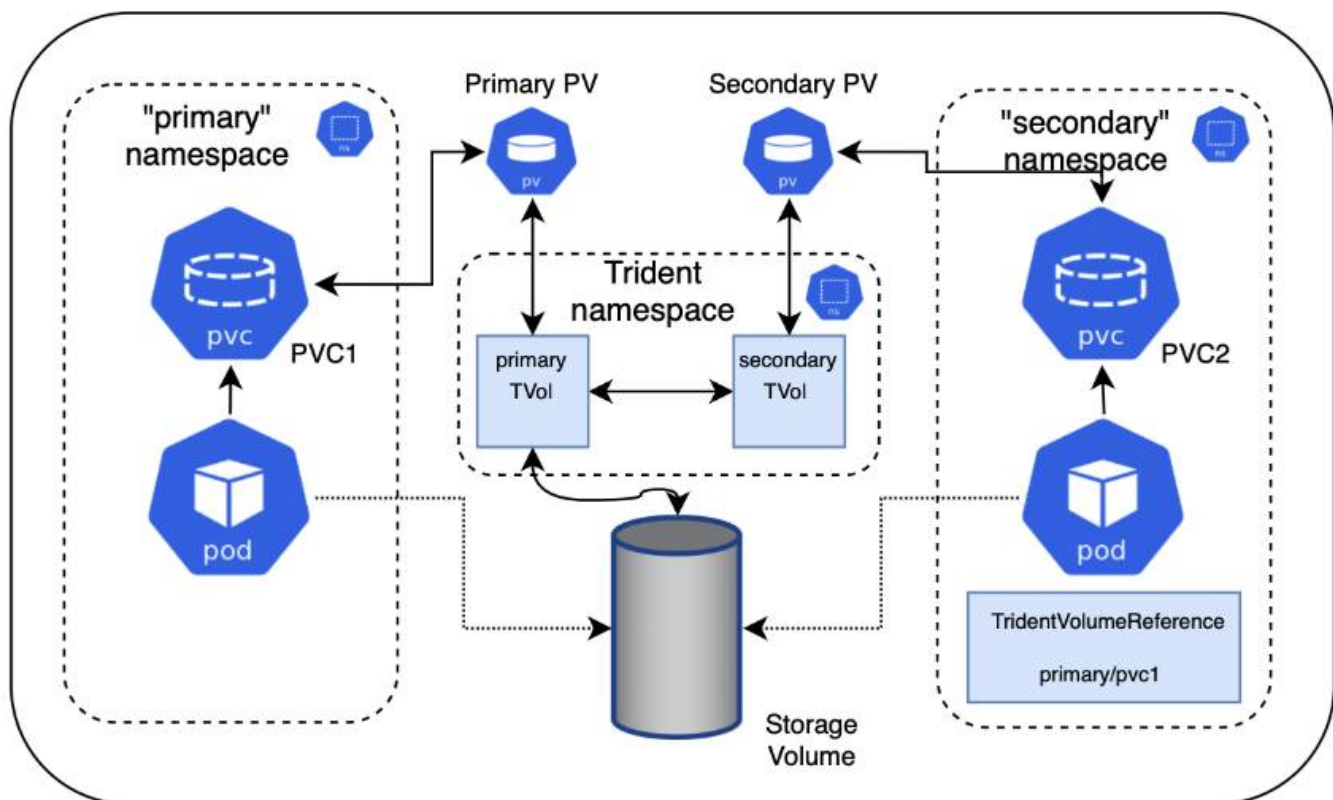
Utilizzando Astra Trident, è possibile creare un volume in uno spazio dei nomi primario e condividerlo in uno o più spazi dei nomi secondari.

### Caratteristiche

Astra TridentVolumeReference CR consente di condividere in modo sicuro volumi NFS ReadWriteMany (RWX) in uno o più spazi dei nomi Kubernetes. Questa soluzione nativa di Kubernetes offre i seguenti vantaggi:

- Diversi livelli di controllo degli accessi per garantire la sicurezza
- Funziona con tutti i driver di volume NFS Trident
- Nessuna dipendenza da tridentctl o da altre funzionalità Kubernetes non native

Questo diagramma illustra la condivisione del volume NFS tra due spazi dei nomi Kubernetes.



### Avvio rapido

Puoi configurare la condivisione dei volumi NFS in pochi passaggi.

1

**Configurare il PVC di origine per la condivisione del volume**

Il proprietario dello spazio dei nomi di origine concede il permesso di accedere ai dati nel PVC di origine.

2

**Concedere il permesso di creare una CR nello spazio dei nomi di destinazione**



L'amministratore del cluster concede l'autorizzazione al proprietario dello spazio dei nomi di destinazione per creare la CR di TridentVolumeReference.

3

### Creare TridentVolumeReference nello spazio dei nomi di destinazione

Il proprietario dello spazio dei nomi di destinazione crea la CR di TridentVolumeReference per fare riferimento al PVC di origine.

4

### Creare il PVC subordinato nello spazio dei nomi di destinazione

Il proprietario dello spazio dei nomi di destinazione crea il PVC subordinato per utilizzare l'origine dati dal PVC di origine.

## Configurare gli spazi dei nomi di origine e di destinazione

Per garantire la sicurezza, la condivisione di spazi dei nomi incrociati richiede la collaborazione e l'azione del proprietario dello spazio dei nomi di origine, dell'amministratore del cluster e del proprietario dello spazio dei nomi di destinazione. Il ruolo dell'utente viene designato in ogni fase.

### Fasi

1. **Source namespace owner:** Crea il PVC (`pvc1`) nello spazio dei nomi di origine che concede l'autorizzazione per la condivisione con lo spazio dei nomi di destinazione (`namespace2`) utilizzando `shareToNamespace` annotazione.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Astra Trident crea il PV e il suo volume di storage NFS back-end.



- È possibile condividere il PVC con più spazi dei nomi utilizzando un elenco delimitato da virgole. Ad esempio, `trident.netapp.io/shareToNamespace: namespace2, namespace3, namespace4`.
- È possibile condividere con tutti gli spazi dei nomi utilizzando `*`. Ad esempio, `trident.netapp.io/shareToNamespace: *`
- È possibile aggiornare il PVC per includere `shareToNamespace` annotazione in qualsiasi momento.

2. **Cluster admin:** creare il ruolo personalizzato e il kubeconfig per concedere l'autorizzazione al proprietario dello spazio dei nomi di destinazione per creare il CR di `TridentVolumeReference` nello spazio dei nomi di destinazione.
3. **Destination namespace owner:** creare una CR di `TridentVolumeReference` nello spazio dei nomi di destinazione che si riferisce allo spazio dei nomi di origine `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Proprietario dello spazio dei nomi di destinazione:** Crea un PVC (`pvc2`) nello spazio dei nomi di destinazione (`namespace2`) utilizzando `shareFromPVC` Annotazione per indicare il PVC di origine.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```



La dimensione del PVC di destinazione deve essere inferiore o uguale al PVC di origine.

## Risultati

Astra Trident legge `shareFromPVC` Annotazione sul PVC di destinazione e crea il PV di destinazione come volume subordinato senza una propria risorsa di storage che punta al PV di origine e condivide la risorsa di storage PV di origine. Il PVC e il PV di destinazione appaiono associati come normali.

## Eliminare un volume condiviso

È possibile eliminare un volume condiviso tra più spazi dei nomi. Astra Trident rimuoverà l'accesso al volume nello spazio dei nomi di origine e manterrà l'accesso ad altri spazi dei nomi che condividono il volume. Una volta rimossi tutti gli spazi dei nomi che fanno riferimento al volume, Astra Trident elimina il volume.

## Utilizzare `tridentctl get` per eseguire query sui volumi subordinati

Utilizzando il `tridentctl` è possibile eseguire `get` comando per ottenere volumi subordinati. Per ulteriori informazioni, fare riferimento al `tridentctl` [comandi e opzioni](#).

```
Usage:
  tridentctl get [option]
```

### Allarmi:

- `-h, --help`: Guida per i volumi.
- `--parentOfSubordinate string`: Limita query al volume di origine subordinato.
- `--subordinateOf string`: Limita la query alle subordinate del volume.

### Limitazioni

- Astra Trident non può impedire la scrittura degli spazi dei nomi di destinazione nel volume condiviso. È necessario utilizzare il blocco dei file o altri processi per impedire la sovrascrittura dei dati dei volumi condivisi.
- Non è possibile revocare l'accesso al PVC di origine rimuovendo `shareToNamespace` oppure `shareFromNamespace` annotazioni o eliminazione di `TridentVolumeReference` CR. Per revocare l'accesso, è necessario eliminare il PVC subordinato.
- Snapshot, cloni e mirroring non sono possibili sui volumi subordinati.

### Per ulteriori informazioni

Per ulteriori informazioni sull'accesso ai volumi tra spazi dei nomi:

- Visitare il sito "[Condivisione di volumi tra spazi dei nomi: Dai il benvenuto all'accesso a volumi tra spazi dei nomi](#)".
- Guarda la demo su "[NetAppTV](#)".

## 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](#)".

## Panoramica

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 essere collocati in diverse zone di disponibilità all'interno di una regione o in diverse regioni. 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 che esegue un ["Versione Kubernetes supportata"](#)

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeadfd99df",
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:"1e11e4a2108024935ecfcb2912226cedeadfd99df",
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"
[nodel,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kuber-
netes.io/arch":"amd64","kubernetes.io/hostname":"nodel","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","kuber-
netes.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","kuber-
netes.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.

Ecco un esempio di definizione di backend:

## YAML

```
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
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
```

## JSON

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "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-centrall
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-centrall
  topology.kubernetes.io/zone: us-centrall-a
- topology.kubernetes.io/region: us-centrall
  topology.kubernetes.io/zone: us-centrall-b
storage:
- labels:
    workload: production
    region: Iowa-DC
    zone: Iowa-DC-A
    supportedTopologies:
    - topology.kubernetes.io/region: us-centrall
      topology.kubernetes.io/zone: us-centrall-a
- labels:
    workload: dev
    region: Iowa-DC
    zone: Iowa-DC-B
    supportedTopologies:
    - topology.kubernetes.io/region: us-centrall
      topology.kubernetes.io/zone: us-centrall-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:



```

kubect1 create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubect1 get pvc
NAME          STATUS      VOLUME      CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san      Pending
2s
kubect1 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
  ----      -
  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: voll
      persistentVolumeClaim:
        claimName: pvc-san
  containers:
    - name: sec-ctx-demo
      image: busybox
      command: [ "sh", "-c", "sleep 1h" ]
      volumeMounts:
        - name: voll
          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

Le snapshot del volume di Kubernetes dei volumi persistenti (PVS) consentono copie point-in-time dei volumi. Puoi creare una snapshot di un volume creato utilizzando Astra Trident, importare uno snapshot creato all'esterno di Astra Trident, creare un nuovo volume da una snapshot esistente e recuperare i dati del volume da snapshot.

### Panoramica

Lo snapshot del volume è supportato da `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, e. `azure-netapp-files` driver.

### Prima di iniziare

Per utilizzare gli snapshot, è necessario disporre di un controller snapshot esterno e di CRD (Custom Resource Definitions). Questa è la responsabilità del Kubernetes orchestrator (ad esempio: Kubeadm, GKE, OpenShift).

Se la distribuzione Kubernetes non include il controller di snapshot e i CRD, fare riferimento a [Implementare un controller per lo snapshot dei volumi](#).



Non creare un controller di snapshot se si creano snapshot di volumi on-demand in un ambiente GKE. GKE utilizza un controller di snapshot integrato e nascosto.

## Creare un'istantanea del volume

### Fasi

1. Creare un `VolumeSnapshotClass`. Per ulteriori informazioni, fare riferimento a ["VolumeSnapshotClass"](#).
  - Il `driver` Indica il driver Astra Trident CSI.
  - `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.

### Esempio

```
cat snap-sc.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

2. Creare un'istantanea di un PVC esistente.

### Esempi

- Questo esempio crea un'istantanea di un PVC esistente.

```
cat snap.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

- Questo esempio crea un oggetto snapshot di volume per un PVC denominato `pvc1` e il nome dello snapshot è impostato su `pvc1-snap`. Un'istantanea `VolumeSnapshot` è analoga a un PVC ed è associata a un `VolumeSnapshotContent` oggetto che rappresenta lo snapshot effettivo.

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

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

- È possibile identificare VolumeSnapshotContent oggetto per pvc1-snap VolumeSnapshot descrivendolo. Il Snapshot Content Name Identifica l'oggetto VolumeSnapshotContent che fornisce questa snapshot. Il Ready To Use Parametro indica che l'istantanea può essere utilizzata per creare un nuovo PVC.

```
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
.
.
```

### Creare un PVC da uno snapshot di volume

È possibile utilizzare dataSource Per creare un PVC utilizzando un VolumeSnapshot denominato <pvc-name> come origine dei dati. Una volta creato, il PVC può essere collegato a un pod e utilizzato come qualsiasi altro PVC.



Il PVC verrà creato nello stesso backend del volume di origine. Fare riferimento a. ["KB: La creazione di un PVC da uno snapshot PVC Trident non può essere creata in un backend alternativo"](#).

Nell'esempio seguente viene creato il PVC utilizzando pvc1-snap come origine dei dati.

```

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: pvcl-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io

```

## Importare uno snapshot di volume

Astra Trident supporta a. ["Processo Snapshot con pre-provisioning di Kubernetes"](#) per consentire all'amministratore del cluster di creare un `VolumeSnapshotContent` Object e importa gli snapshot creati all'esterno di Astra Trident.

### Prima di iniziare

Astra Trident deve aver creato o importato il volume principale dello snapshot.

### Fasi

1. **Cluster admin:** creare un `VolumeSnapshotContent` oggetto che fa riferimento allo snapshot backend. In questo modo viene avviato il flusso di lavoro delle snapshot in Astra Trident.
  - Specificare il nome dell'istantanea backend in annotations come `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
  - Specificare `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` poll `snapshotHandle`. Queste sono le uniche informazioni fornite a Astra Trident dallo snap-ter esterno in `ListSnapshots` chiamata.



Il `<volumeSnapshotContentName>` impossibile corrispondere sempre al nome dell'istantanea backend a causa di vincoli di denominazione CR.

### Esempio

Nell'esempio seguente viene creato un `VolumeSnapshotContent` oggetto che fa riferimento allo snapshot backend `snap-01`.

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01" # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>

```

- Cluster admin:** creare il `VolumeSnapshot` CR che fa riferimento a `VolumeSnapshotContent` oggetto. In questo modo viene richiesto l'accesso per l'utilizzo di `VolumeSnapshot` in un determinato namespace.

### Esempio

Nell'esempio seguente viene creato un `VolumeSnapshot` CR con nome `import-snap` questo fa riferimento al `VolumeSnapshotContent` con nome `import-snap-content`.

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content

```

- Elaborazione interna (nessuna azione richiesta):** lo snapshot esterno riconosce la nuova creazione `VolumeSnapshotContent` ed esegue `ListSnapshots` chiamata. Astra Trident crea l'`TridentSnapshot`.
  - Lo snapshot esterno imposta `VolumeSnapshotContent` a `readyToUse` e a `VolumeSnapshot` a `true`.
  - Trident ritorna `readyToUse=true`.
- Qualsiasi utente:** creare un `PersistentVolumeClaim` per fare riferimento al nuovo `VolumeSnapshot`, dove il `spec.dataSource` (o `spec.dataSourceRef`) è il `VolumeSnapshot` nome.

### Esempio

Nell'esempio seguente viene creato un PVC che fa riferimento a `VolumeSnapshot` con nome `import-snap`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

### Ripristinare i dati del volume utilizzando le snapshot

La directory Snapshot è nascosta per impostazione predefinita per facilitare la massima compatibilità dei volumi con cui viene eseguito il provisioning mediante `ontap-nas` e `ontap-nas-economy` driver. Attivare il `.snapshot` directory per ripristinare i dati direttamente dalle snapshot.

Utilizzare la CLI ONTAP per il ripristino dello snapshot del volume per ripristinare uno stato di un volume registrato in uno snapshot precedente.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```



Quando si ripristina una copia snapshot, la configurazione del volume esistente viene sovrascritta. Le modifiche apportate ai dati del volume dopo la creazione della copia snapshot andranno perse.

### Eliminare un PV con gli snapshot associati

Quando si elimina un volume persistente con snapshot associate, il volume Trident corrispondente viene aggiornato a uno stato di eliminazione. Rimuovere le snapshot del volume per eliminare il volume Astra Trident.

### Implementare un controller per lo snapshot dei volumi

Se la distribuzione Kubernetes non include lo snapshot controller e i CRD, è possibile implementarli come segue.

#### Fasi

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-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

## 2. Creare il controller di snapshot.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```



Se necessario, aprire `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e aggiornare namespace allo spazio dei nomi.

### Link correlati

- ["Snapshot dei volumi"](#)
- ["VolumeSnapshotClass"](#)

## Informazioni sul copyright

Copyright © 2024 NetApp, Inc. Tutti i diritti riservati. Stampato negli Stati Uniti d'America. Nessuna porzione di questo documento soggetta a copyright può essere riprodotta in qualsiasi formato o mezzo (grafico, elettronico o meccanico, inclusi fotocopie, registrazione, nastri o storage in un sistema elettronico) senza previo consenso scritto da parte del detentore del copyright.

Il software derivato dal materiale sottoposto a copyright di NetApp è soggetto alla seguente licenza e dichiarazione di non responsabilità:

IL PRESENTE SOFTWARE VIENE FORNITO DA NETAPP "COSÌ COM'È" E SENZA QUALSIVOGLIA TIPO DI GARANZIA IMPLICITA O ESPRESSA FRA CUI, A TITOLO ESEMPLIFICATIVO E NON ESAUSTIVO, GARANZIE IMPLICITE DI COMMERCIALIZZABILITÀ E IDONEITÀ PER UNO SCOPO SPECIFICO, CHE VENGONO DECLINATE DAL PRESENTE DOCUMENTO. NETAPP NON VERRÀ CONSIDERATA RESPONSABILE IN ALCUN CASO PER QUALSIVOGLIA DANNO DIRETTO, INDIRETTO, ACCIDENTALE, SPECIALE, ESEMPLARE E CONSEGUENZIALE (COMPRESI, A TITOLO ESEMPLIFICATIVO E NON ESAUSTIVO, PROCUREMENT O SOSTITUZIONE DI MERCI O SERVIZI, IMPOSSIBILITÀ DI UTILIZZO O PERDITA DI DATI O PROFITTI OPPURE INTERRUZIONE DELL'ATTIVITÀ AZIENDALE) CAUSATO IN QUALSIVOGLIA MODO O IN RELAZIONE A QUALUNQUE TEORIA DI RESPONSABILITÀ, SIA ESSA CONTRATTUALE, RIGOROSA O DOVUTA A INSOLVENZA (COMPRESA LA NEGLIGENZA O ALTRO) INSORTA IN QUALSIASI MODO ATTRAVERSO L'UTILIZZO DEL PRESENTE SOFTWARE ANCHE IN PRESENZA DI UN PREAVVISO CIRCA L'EVENTUALITÀ DI QUESTO TIPO DI DANNI.

NetApp si riserva il diritto di modificare in qualsiasi momento qualunque prodotto descritto nel presente documento senza fornire alcun preavviso. NetApp non si assume alcuna responsabilità circa l'utilizzo dei prodotti o materiali descritti nel presente documento, con l'eccezione di quanto concordato espressamente e per iscritto da NetApp. L'utilizzo o l'acquisto del presente prodotto non comporta il rilascio di una licenza nell'ambito di un qualche diritto di brevetto, marchio commerciale o altro diritto di proprietà intellettuale di NetApp.

Il prodotto descritto in questa guida può essere protetto da uno o più brevetti degli Stati Uniti, esteri o in attesa di approvazione.

LEGENDA PER I DIRITTI SOTTOPOSTI A LIMITAZIONE: l'utilizzo, la duplicazione o la divulgazione da parte degli enti governativi sono soggetti alle limitazioni indicate nel sottoparagrafo (b)(3) della clausola Rights in Technical Data and Computer Software del DFARS 252.227-7013 (FEB 2014) e FAR 52.227-19 (DIC 2007).

I dati contenuti nel presente documento riguardano un articolo commerciale (secondo la definizione data in FAR 2.101) e sono di proprietà di NetApp, Inc. Tutti i dati tecnici e il software NetApp forniti secondo i termini del presente Contratto sono articoli aventi natura commerciale, sviluppati con finanziamenti esclusivamente privati. Il governo statunitense ha una licenza irrevocabile limitata, non esclusiva, non trasferibile, non cedibile, mondiale, per l'utilizzo dei Dati esclusivamente in connessione con e a supporto di un contratto governativo statunitense in base al quale i Dati sono distribuiti. Con la sola esclusione di quanto indicato nel presente documento, i Dati non possono essere utilizzati, divulgati, riprodotti, modificati, visualizzati o mostrati senza la previa approvazione scritta di NetApp, Inc. I diritti di licenza del governo degli Stati Uniti per il Dipartimento della Difesa sono limitati ai diritti identificati nella clausola DFARS 252.227-7015(b) (FEB 2014).

## Informazioni sul marchio commerciale

NETAPP, il logo NETAPP e i marchi elencati alla pagina <http://www.netapp.com/TM> sono marchi di NetApp, Inc. Gli altri nomi di aziende e prodotti potrebbero essere marchi dei rispettivi proprietari.