# Use o Astra Trident

## Astra Trident

NetApp
February 05, 2025

# Índice

# Use o Astra Trident

## Configurar backends

Um back-end define a relação entre o Astra Trident e um sistema de storage. Ele diz ao Astra Trident como se comunicar com esse sistema de storage e como o Astra Trident deve provisionar volumes a partir dele. O Astra Trident oferecerá automaticamente pools de storage de back-ends que atendem aos requisitos definidos por uma classe de storage. Saiba mais sobre como configurar o back-end com base no tipo de sistema de armazenamento que você tem.

- "Configurar um back-end do Azure NetApp Files"
- "Configure um back-end do Cloud Volumes Service para o Google Cloud Platform"
- "Configurar um back-end NetApp HCI ou SolidFire"
- "Configurar um back-end com drivers nas ONTAP ou Cloud Volumes ONTAP"
- "Configure um back-end com drivers SAN ONTAP ou Cloud Volumes ONTAP"
- "Use o Astra Trident com o Amazon FSX para NetApp ONTAP"

### Configurar um back-end do Azure NetApp Files

Saiba mais sobre como configurar o Azure NetApp Files (ANF) como back-end para sua instalação do Astra Trident usando as configurações de amostra fornecidas.

> O serviço Azure NetApp Files não suporta volumes inferiores a 100 GB. O Astra Trident cria automaticamente volumes de 100 GB se um volume menor for solicitado.

#### O que você vai precisar

Para configurar e usar um "Azure NetApp Files" back-end, você precisa do seguinte:

- `subscriptionID` A partir de uma subscrição do Azure com o Azure NetApp Files ativado.
- `tenantID`, `clientID` E `clientSecret` de um "Registo da aplicação" no Azure ative Directory com permissões suficientes para o serviço Azure NetApp Files. O Registo de aplicações deve utilizar a `Owner` função ou `Contributor` predefinida pelo Azure.

> Para saber mais sobre as funções incorporadas do Azure, consulte o "Documentação do Azure".

- O Azure `location` que contém pelo menos um "sub-rede delegada". A partir do Trident 22,01, o `location` parâmetro é um campo obrigatório no nível superior do arquivo de configuração de back-end. Os valores de localização especificados em pools virtuais são ignorados.
- Se você estiver usando o Azure NetApp Files pela primeira vez ou em um novo local, alguma configuração inicial será necessária. Consulte "guia quickstart" .

#### Sobre esta tarefa

Com base na configuração de back-end (sub-rede, rede virtual, nível de serviço e local), o Trident cria volumes do ANF em pools de capacidade disponíveis no local solicitado e correspondem ao nível de serviço e à sub-rede solicitados.

OBSERVAÇÃO: O Astra Trident não é compatível com pools de capacidade de QoS manual.

### Opções de configuração de back-end

Consulte a tabela a seguir para obter as opções de configuração de back-end:

| Parâmetro | Descrição | Padrão |
|---|---|---|
| `version` | | Sempre 1 |
| `storageDriverName` | Nome do controlador de armazenamento | "ficheiros azure-NetApp" |
| `backendName` | Nome personalizado ou back-end de storage | Nome do condutor e carateres aleatórios |
| `subscriptionID` | O ID da assinatura da sua assinatura do Azure | |
| `tenantID` | O ID do locatário de um Registro de aplicativo | |
| `clientID` | A ID do cliente de um registo de aplicação | |
| `clientSecret` | O segredo do cliente de um Registro de aplicativo | |
| `serviceLevel` | Um de `Standard`, `Premium`, ou `Ultra` | "" (aleatório) |
| `location` | Nome do local do Azure onde os novos volumes serão criados | |
| `serviceLevel` | Um de `Standard`, `Premium`, ou `Ultra` | "" (aleatório) |
| `resourceGroups` | Lista de grupos de recursos para filtragem de recursos descobertos | "[]" (sem filtro) |
| `netappAccounts` | Lista de contas do NetApp para filtragem de recursos descobertos | "[]" (sem filtro) |
| `capacityPools` | Lista de pools de capacidade para filtrar recursos descobertos | "[]" (sem filtro, aleatório) |
| `virtualNetwork` | Nome de uma rede virtual com uma sub-rede delegada | "" |
| `subnet` | Nome de uma sub-rede delegada `Microsoft.Netapp/volumes` | "" |
| `nfsMountOptions` | Controle refinado das opções de montagem NFS. | "3" |
| `limitVolumeSize` | Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor | "" (não aplicado por padrão) |

| Parâmetro | Descrição | Padrão |
|---|---|---|
| `debugTraceFlags` | Debug flags para usar ao solucionar problemas. Exemplo, `\{"api": false, "method": true, "discovery": true}`. Não use isso a menos que você esteja solucionando problemas e exija um despejo de log detalhado. | nulo |

Se você encontrar um erro "sem pools de capacidade encontrados" ao tentar criar um PVC, é provável que o Registro do aplicativo não tenha as permissões e recursos necessários (sub-rede, rede virtual, pool de capacidade) associados. O Astra Trident registrará os recursos do Azure descobertos quando o back-end for criado quando o debug estiver habilitado. Certifique-se de verificar se está a ser utilizada uma função adequada.

Se você quiser montar os volumes usando o NFS versão 4,1, você pode incluir `nfsvers=4` na lista de opções de montagem delimitadas por vírgulas para escolher NFS v4,1. Todas as opções de montagem definidas em uma classe de armazenamento substituem as opções de montagem definidas em um arquivo de configuração de back-end.

Os valores para `resourceGroups`, `netappAccounts`, `capacityPools`, , `virtualNetwork` e `subnet` podem ser especificados usando nomes curtos ou totalmente qualificados. Nomes curtos podem corresponder vários recursos com o mesmo nome, portanto, o uso de nomes totalmente qualificados é recomendado na maioria das situações. Os `resourceGroups` valores , `netappAccounts`, e `capacityPools` são filtros que restringem o conjunto de recursos descobertos aos disponíveis para esse back-end de armazenamento e podem ser especificados em qualquer combinação. Os nomes totalmente qualificados são do seguinte formato:

| Tipo | Formato |
|---|---|
| Grupo de recursos | <resource group> |
| Conta NetApp | <resource group>/ cliente NetApp account> |
| Pool de capacidade | <resource group>/ cliente NetApp account>/<capacity pool> |
| Rede virtual | <resource group>/<virtual network> |
| Sub-rede | <resource group>/<virtual network>/<subnet> |

Você pode controlar como cada volume é provisionado por padrão, especificando as seguintes opções em uma seção especial do arquivo de configuração. Veja os exemplos de configuração abaixo.

| Parâmetro | Descrição | Padrão |
|---|---|---|
| `exportRule` | As regras de exportação para novos volumes | "0,0.0,0/0" |
| `snapshotDir` | Controla a visibilidade do diretório .snapshot | "falso" |
| `size` | O tamanho padrão dos novos volumes | "100G" |

| Parâmetro | Descrição | Padrão |
| --- | --- | --- |
| `unixPermissions` | As permissões unix de novos volumes (4 dígitos octal) | "" (recurso de pré-visualização, requer lista branca na assinatura) |

O `exportRule` valor deve ser uma lista separada por vírgulas de qualquer combinação de endereços IPv4 ou sub-redes IPv4 na notação CIDR.

> Para todos os volumes criados em um back-end do ANF, o Astra Trident copia todas as etiquetas presentes em um pool de storage para o volume de storage no momento em que ele é provisionado. Os administradores de storage podem definir rótulos por pool de storage e agrupar todos os volumes criados em um pool de storage. Isso fornece uma maneira conveniente de diferenciar volumes com base em um conjunto de rótulos personalizáveis que são fornecidos na configuração de back-end.

**Exemplo 1: Configuração mínima**

Esta é a configuração mínima absoluta de back-end. Com essa configuração, o Astra Trident descobre todas as suas contas NetApp, pools de capacidade e sub-redes delegadas no ANF no local configurado e coloca novos volumes aleatoriamente em um desses pools e sub-redes.

Essa configuração é ideal quando você está apenas começando o ANF e experimentando as coisas, mas na prática você vai querer fornecer um escopo adicional para os volumes provisionados.

```
{
    "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"
}
```

**Exemplo 2: Configuração específica de nível de serviço com filtros de pool de capacidade**

Essa configuração de back-end coloca volumes no local do Azure `eastus` em um `Ultra` pool de capacidade. O Astra Trident descobre automaticamente todas as sub-redes delegadas no ANF nesse local e coloca um novo volume em uma delas aleatoriamente.

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

**Exemplo 3: Configuração avançada**

Essa configuração de back-end reduz ainda mais o escopo do posicionamento de volume para uma única sub-rede e também modifica alguns padrões de provisionamento de 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",
        "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"
=======
        }
    }
```

**Exemplo 4: Configuração do pool de armazenamento virtual**

Essa configuração de back-end define vários pools de storage em um único arquivo. Isso é útil quando você tem vários pools de capacidade com suporte a diferentes níveis de serviço e deseja criar classes de storage no Kubernetes que os representem.

```
{
    "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"],
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
    "labels": {
        "cloud": "azure"
    },
    "location": "eastus",

    "storage": [
        {
            "labels": {
                "performance": "gold"
            },
            "serviceLevel": "Ultra",
            "capacityPools": ["ultra-1", "ultra-2"]
        },
        {
            "labels": {
                "performance": "silver"
            },
            "serviceLevel": "Premium",
            "capacityPools": ["premium-1"]
        },
        {
            "labels": {
                "performance": "bronze"
            },
            "serviceLevel": "Standard",
            "capacityPools": ["standard-1", "standard-2"]
        }
    ]
}
```

As definições a seguir `StorageClass` referem-se aos pools de armazenamento acima. Ao usar o `parameters.selector` campo, você pode especificar para cada `StorageClass` um o pool visrtual que é usado para hospedar um volume. O volume terá os aspetos definidos no pool escolhido.

```
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
```

**O que se segue?**

Depois de criar o arquivo de configuração de back-end, execute o seguinte comando:

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

Se a criação do backend falhar, algo está errado com a configuração do backend. Você pode exibir os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs
```

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode executar o comando create novamente.

## Configurar um back-end do CVS para GCP

Saiba como configurar o NetApp Cloud Volumes Service (CVS) para o Google Cloud Platform (GCP) como

back-end para a instalação do Astra Trident usando as configurações de exemplo fornecidas.

> O NetApp Cloud Volumes Service não é compatível com volumes CVS-performance com tamanho inferior a 100 GiB ou volumes CVS com tamanho inferior a 300 GiB. O Astra Trident cria automaticamente volumes do tamanho mínimo se a o volume solicitado for menor que o tamanho mínimo.

**O que você vai precisar**

Para configurar e usar o "Cloud Volumes Service para Google Cloud" back-end, você precisa do seguinte:

- Uma conta do Google Cloud configurada com o NetApp CVS
- Número do projeto da sua conta do Google Cloud
- Conta de serviço do Google Cloud com a `netappcloudvolumes.admin` função
- Arquivo de chave de API para sua conta de serviço CVS

O Astra Trident agora inclui suporte a volumes menores com o padrão "Tipo de serviço CVS no GCP". Para backends criados com `storageClass=software`o , os volumes agora terão um tamanho mínimo de provisionamento de 300 GiB. O CVS atualmente fornece esse recurso sob disponibilidade controlada e não fornece suporte técnico. Os usuários devem se inscrever para acesso a volumes menores de 1TiB "aqui" TB . A NetApp recomenda que os clientes consumam volumes inferiores a 1TiB TB para **cargas de trabalho que não sejam de produção**.

> Ao implantar backends usando o tipo de serviço CVS padrão (`storageClass=software`), os usuários devem obter acesso ao recurso volumes sub-1TiB no GCP para o(s) número(s) de Projeto e ID(s) de Projeto em questão. Isso é necessário para que o Astra Trident provisione volumes inferiores a 1TiB TB. Caso contrário, as criações de volume falharão para PVCs menores que 600 GiB. Obter acesso a volumes inferiores a 1TiB com "este formulário"o .

Os volumes criados pelo Astra Trident para o nível de serviço CVS padrão serão provisionados da seguinte forma:

- PVCs menores que 300 GiB resultarão em Astra Trident criando um volume CVS de 300 GiB.
- Os PVCs que estão entre 300 GiB e 600 GiB resultarão na criação do Astra Trident de um volume CVS do tamanho solicitado.
- Os PVCs que estão entre 600 GiB e 1 TIB resultarão na criação de um volume CVS de 1TiB TB do Astra Trident.
- PVCs maiores que 1 TIB resultarão na criação do Astra Trident de um volume CVS do tamanho solicitado.

**Opções de configuração de back-end**

Consulte a tabela a seguir para obter as opções de configuração de back-end:

| Parâmetro | Descrição | Padrão |
|---|---|---|
| `version` | | Sempre 1 |
| `storageDriverName` | Nome do controlador de armazenamento | "gcp-cvs" |
| `backendName` | Nome personalizado ou back-end de storage | Nome do driver e parte da chave da API |

| Parâmetro | Descrição | Padrão |
| --- | --- | --- |
| `storageClass` | Tipo de armazenamento. Escolha entre `hardware` (otimizado para performance) ou `software` (tipo de serviço CVS) | |
| `projectNumber` | Número do projeto da conta Google Cloud. O valor é encontrado na página inicial do portal do Google Cloud. | |
| `apiRegion` | Região da conta CVS. É a região onde o backend provisionará os volumes. | |
| `apiKey` | Chave de API para a conta de serviço do Google Cloud com a `netappcloudvolumes.admin` função. Ele inclui o conteúdo formatado em JSON do arquivo de chave privada de uma conta de serviço do Google Cloud (copiado literalmente no arquivo de configuração de back-end). | |
| `proxyURL` | URL do proxy se o servidor proxy for necessário para se conetar à conta CVS. O servidor proxy pode ser um proxy HTTP ou um proxy HTTPS. Para um proxy HTTPS, a validação do certificado é ignorada para permitir o uso de certificados autoassinados no servidor proxy. Os servidores proxy com autenticação ativada não são suportados. | |
| `nfsMountOptions` | Controle refinado das opções de montagem NFS. | "3" |
| `limitVolumeSize` | Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor | "" (não aplicado por padrão) |
| `serviceLevel` | O nível de serviço CVS para novos volumes. Os valores são "padrão", "premium" e "extremo". | "standard" (padrão) |
| `network` | Rede GCP usada para volumes CVS | "padrão" |

| Parâmetro | Descrição | Padrão |
| --- | --- | --- |
| `debugTraceFlags` | Debug flags para usar ao solucionar problemas. Exemplo, `\{"api":false, "method":true}`. Não use isso a menos que você esteja solucionando problemas e exija um despejo de log detalhado. | nulo |

Se estiver usando uma rede VPC compartilhada, ambos `projectNumber` e `hostProjectNumber` devem ser especificados. Nesse caso, `projectNumber` é o projeto de serviço, e `hostProjectNumber` é o projeto host.

O `apiRegion` representa a região do GCP em que o Astra Trident cria volumes CVS. Ao criar clusters de Kubernetes entre regiões, os volumes CVS criados em um `apiRegion` podem ser usados em workloads programados em nós em várias regiões do GCP. Esteja ciente de que o tráfego entre regiões incorre em um custo adicional.

- Para habilitar o acesso entre regiões, a definição do StorageClass para `allowedTopologies` deve incluir todas as regiões. Por exemplo:

```
- key: topology.kubernetes.io/region
  values:
  - us-east1
  - europe-west1
```

- `storageClass` é um parâmetro opcional que você pode usar para selecionar o desejado "Tipo de serviço CVS". Você pode escolher entre o tipo de serviço CVS básico (`storageClass=software`) ou o tipo de serviço CVS-Performance (`storageClass=hardware`), que o Trident usa por padrão. Certifique-se de especificar um `apiRegion` que forneça o CVS respetivo `storageClass` na definição de back-end.

A integração do Astra Trident com o tipo de serviço CVS básico no Google Cloud é um recurso **beta**, não destinado a cargas de trabalho de produção. O Trident é **totalmente suportado** com o tipo de serviço CVS-Performance e o usa por padrão.

Cada back-end provisiona volumes em uma única região do Google Cloud. Para criar volumes em outras regiões, você pode definir backends adicionais.

Você pode controlar como cada volume é provisionado por padrão, especificando as seguintes opções em uma seção especial do arquivo de configuração. Veja os exemplos de configuração abaixo.

| Parâmetro | Descrição | Padrão |
| --- | --- | --- |
| `exportRule` | As regras de exportação para novos volumes | "0,0.0,0/0" |
| `snapshotDir` | Acesso ao `.snapshot` diretório | "falso" |

| Parâmetro | Descrição | Padrão |
|---|---|---|
| `snapshotReserve` | Porcentagem de volume reservado para snapshots | "" (aceitar o padrão CVS de 0) |
| `size` | O tamanho dos novos volumes | "100Gi" |

O `exportRule` valor deve ser uma lista separada por vírgulas de qualquer combinação de endereços IPv4 ou sub-redes IPv4 na notação CIDR.

Para todos os volumes criados em um back-end do Google Cloud do CVS, o Trident copia todas as etiquetas presentes em um pool de storage para o volume de storage no momento em que ele é provisionado. Os administradores de storage podem definir rótulos por pool de storage e agrupar todos os volumes criados em um pool de storage. Isso fornece uma maneira conveniente de diferenciar volumes com base em um conjunto de rótulos personalizáveis que são fornecidos na configuração de back-end.

**Exemplo 1: Configuração mínima**

Esta é a configuração mínima absoluta de back-end.

```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "apiRegion": "us-west2",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "12345678901234567890123456789012345678901234567890",
        "private_key": "
        -----BEGIN PRIVATE KEY-----
        <key_value>
        -----END PRIVATE KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    }
}
```

**Exemplo 2: Configuração do tipo de serviço CVS básico**

Este exemplo mostra uma definição de back-end que usa o tipo de serviço CVS básico, destinado a cargas de trabalho de uso geral e fornece desempenho leve/moderado, juntamente com alta disponibilidade por zona.

```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "storageClass": "software",
    "apiRegion": "us-east4",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "<id_value>>",
        "private_key": "
        -----BEGIN PRIVATE KEY-----
        <key_value>
        -----END PRIVATE KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    }
}
```

**Exemplo 3: Configuração de nível de serviço único**

Este exemplo mostra um arquivo de back-end que aplica os mesmos aspectos a todo o storage criado pelo Astra Trident na região Google Cloud US-west2. Este exemplo também mostra o uso do `proxyURL` no arquivo de configuração de back-end.

```
{
    "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-----
        <key_value>
        -----END PRIVATE KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    },
    "proxyURL": "http://proxy-server-hostname/",
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
    "limitVolumeSize": "10Ti",
    "serviceLevel": "premium",
    "defaults": {
        "snapshotDir": "true",
        "snapshotReserve": "5",
        "exportRule": "10.0.0.0/24,10.0.1.0/24,10.0.2.100",
        "size": "5Ti"
    }
}
```

**Exemplo 4: Configuração do pool de armazenamento virtual**

Este exemplo mostra o arquivo de definição de back-end configurado com pools de armazenamento virtual juntamente com `StorageClasses` isso se referem a eles.

No arquivo de definição de back-end de exemplo mostrado abaixo, padrões específicos são definidos para todos os pools de armazenamento, que definem o `snapshotReserve` em 5% e o `exportRule` para 0,0.0,0/0. Os pools de armazenamento virtual são definidos na `storage` seção. Neste exemplo, cada pool de armazenamento individual define seu próprio `serviceLevel`, e alguns pools substituem os valores padrão.

```
{
    "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-----
        <key_value>
        -----END PRIVATE KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    },
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",

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

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

    "storage": [
        {
            "labels": {
                "performance": "extreme",
                "protection": "extra"
            },
            "serviceLevel": "extreme",
            "defaults": {
                "snapshotDir": "true",
                "snapshotReserve": "10",
```

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

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

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

As seguintes definições do StorageClass referem-se aos pools de armazenamento acima. Usando o `parameters.selector` campo, você pode especificar para cada StorageClass o pool virtual usado para hospedar um volume. O volume terá os aspetos definidos no pool escolhido.

O primeiro StorageClass ) (`cvs-extreme-extra-protection`mapeia para o primeiro pool de armazenamento virtual. Esse é o único pool que oferece desempenho extremo com uma reserva de snapshot de 10%. O último StorageClass ) (`cvs-extra-protection`chama qualquer pool de armazenamento que forneça uma reserva de snapshot de 10%. O Astra Trident decide qual pool de storage virtual está selecionado e garante

que o requisito de reserva de snapshot seja atendido.

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

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

**O que se segue?**

Depois de criar o arquivo de configuração de back-end, execute o seguinte comando:

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

Se a criação do backend falhar, algo está errado com a configuração do backend. Você pode exibir os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs
```

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode executar o comando create novamente.

## Configurar um back-end NetApp HCI ou SolidFire

Saiba mais sobre como criar e usar um back-end Element com sua instalação do Astra Trident.

**O que você vai precisar**

- Um sistema de storage compatível que executa o software Element.
- Credenciais para um usuário de administrador ou locatário de cluster do NetApp HCI/SolidFire que possa gerenciar volumes.
- Todos os seus nós de trabalho do Kubernetes devem ter as ferramentas iSCSI apropriadas instaladas. "informações sobre a preparação do nó de trabalho"Consulte .

**O que você precisa saber**

O `solidfire-san` driver de armazenamento suporta ambos os modos de volume: Arquivo e bloco. Para o `Filesystem` volumeMode, o Astra Trident cria um volume e cria um sistema de arquivos. O tipo de sistema de arquivos é especificado pelo StorageClass.

| Condutor | Protocolo | Modo de volume | Modos de acesso suportados | Sistemas de arquivos suportados |
|---|---|---|---|---|
| `solidfire-san` | ISCSI | Bloco | RWO, ROX, RWX | Sem sistema de ficheiros. Dispositivo de bloco bruto. |
| `solidfire-san` | ISCSI | Bloco | RWO, ROX, RWX | Sem sistema de ficheiros. Dispositivo de bloco bruto. |
| `solidfire-san` | ISCSI | Sistema de ficheiros | RWO, ROX | `xfs` `ext3`, , `ext4` |
| `solidfire-san` | ISCSI | Sistema de ficheiros | RWO, ROX | `xfs` `ext3`, , `ext4` |

O Astra Trident usa o CHAP quando funciona como um supervisor de CSI aprimorado. Se você estiver usando CHAP (que é o padrão para CSI), nenhuma preparação adicional é necessária. Recomenda-se definir explicitamente a `UseCHAP` opção para usar CHAP com Trident não-CSI. Caso contrário, "aqui"consulte .

Os grupos de acesso a volume só são compatíveis com a estrutura convencional não CSI para Astra Trident. Quando configurado para funcionar no modo CSI, o Astra Trident usa CHAP.

Se nenhuma `AccessGroups` ou `UseCHAP` for definida, uma das seguintes regras será aplicada:

- Se o grupo de acesso padrão `trident` for detetado, os grupos de acesso serão usados.
- Se nenhum grupo de acesso for detetado e a versão do Kubernetes for 1,7 ou posterior, o CHAP será usado.

### Opções de configuração de back-end

Consulte a tabela a seguir para obter as opções de configuração de back-end:

| Parâmetro | Descrição | Padrão |
|---|---|---|
| `version` | | Sempre 1 |
| `storageDriverName` | Nome do controlador de armazenamento | Sempre "SolidFire-san" |
| `backendName` | Nome personalizado ou back-end de storage | Endereço IP "SolidFire_" e armazenamento (iSCSI) |
| `Endpoint` | MVIP para o cluster SolidFire com credenciais de locatário | |
| `SVIP` | Porta e endereço IP de armazenamento (iSCSI) | |

| Parâmetro | Descrição | Padrão |
| --- | --- | --- |
| `labels` | Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes. | "" |
| `TenantName` | Nome do locatário a utilizar (criado se não for encontrado) | |
| `InitiatorIFace` | Restringir o tráfego iSCSI a uma interface de host específica | "padrão" |
| `UseCHAP` | Use CHAP para autenticar iSCSI | verdadeiro |
| `AccessGroups` | Lista de IDs de Grupo de Acesso a utilizar | Encontra a ID de um grupo de acesso chamado "Trident" |
| `Types` | Especificações de QoS | |
| `limitVolumeSize` | Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor | "" (não aplicado por padrão) |
| `debugTraceFlags` | Debug flags para usar ao solucionar problemas. Por exemplo, "api":false, "método":true" | nulo |

Não use `debugTraceFlags` a menos que você esteja solucionando problemas e exija um despejo de log detalhado.

Para todos os volumes criados, o Astra Trident copiará todas as etiquetas presentes em um pool de storage para a LUN de storage de backup no momento em que ela for provisionada. Os administradores de storage podem definir rótulos por pool de storage e agrupar todos os volumes criados em um pool de storage. Isso fornece uma maneira conveniente de diferenciar volumes com base em um conjunto de rótulos personalizáveis que são fornecidos na configuração de back-end.

**Exemplo 1: Configuração de back-end para `solidfire-san` driver com três tipos de volume**

Este exemplo mostra um arquivo de back-end usando autenticação CHAP e modelagem de três tipos de volume com garantias de QoS específicas. Provavelmente você definiria classes de armazenamento para consumir cada uma delas usando o `IOPS` parâmetro de classe de armazenamento.

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

**Exemplo 2: Configuração de classe de back-end e armazenamento para `solidfire-san` driver com pools de armazenamento virtual**

Este exemplo mostra o arquivo de definição de back-end configurado com pools de armazenamento virtual junto com o StorageClasses que se referem a eles.

No arquivo de definição de back-end de exemplo mostrado abaixo, padrões específicos são definidos para todos os pools de armazenamento, que definem o `type` em Prata. Os pools de armazenamento virtual são definidos na `storage` seção. Neste exemplo, alguns conjuntos de armazenamento definem seu próprio tipo e alguns conjuntos substituem os valores padrão definidos acima.

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

As seguintes definições do StorageClass referem-se aos pools de armazenamento virtual acima. Usando o `parameters.selector` campo, cada StorageClass chama qual(s) pool(s) virtual(s) pode(m) ser(ão) usado(s) para hospedar um volume. O volume terá os aspetos definidos no pool virtual escolhido.

O primeiro StorageClass ) (`solidfire-gold-four`será mapeado para o primeiro pool de

armazenamento virtual. Este é o único pool que oferece desempenho de ouro com um `Volume Type QoS` de ouro. O último StorageClass ) (`solidfire-silver`chama qualquer pool de armazenamento que ofereça um desempenho prateado. O Astra Trident decidirá qual pool de storage virtual está selecionado e garantirá que o requisito de storage seja atendido.

```
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"
```

### Encontre mais informações

- "Grupos de acesso de volume"

## Configure um back-end com drivers SAN ONTAP ou Cloud Volumes ONTAP

Saiba mais sobre como configurar um back-end ONTAP com drivers SAN ONTAP e Cloud Volumes ONTAP.

- "Preparação"
- "Configuração e exemplos"

### Permissões do usuário

O Astra Trident espera ser executado como administrador da ONTAP ou SVM, normalmente usando o `admin` usuário do cluster ou um `vsadmin` usuário SVM, ou um usuário com um nome diferente que tenha a mesma função. Para implantações do Amazon FSX for NetApp ONTAP, o Astra Trident espera ser executado como administrador do ONTAP ou SVM, usando o usuário do cluster `fsxadmin` ou um `vsadmin` usuário SVM, ou um usuário com um nome diferente que tenha a mesma função. O `fsxadmin` usuário é um substituto limitado para o usuário administrador do cluster.

> Se você usar o `limitAggregateUsage` parâmetro, as permissões de administrador do cluster serão necessárias. Ao usar o Amazon FSX for NetApp ONTAP com Astra Trident, o `limitAggregateUsage` parâmetro não funcionará com as `vsadmin` contas de usuário e `fsxadmin`. A operação de configuração falhará se você especificar este parâmetro.

Embora seja possível criar uma função mais restritiva no ONTAP que um driver Trident pode usar, não recomendamos. A maioria das novas versões do Trident chamarão APIs adicionais que teriam que ser contabilizadas, tornando as atualizações difíceis e suscetíveis a erros.

### Preparação

Saiba mais sobre como se preparar para configurar um back-end ONTAP com drivers SAN ONTAP. Para todos os back-ends ONTAP, o Astra Trident requer pelo menos um agregado atribuído ao SVM.

Lembre-se de que você também pode executar mais de um driver e criar classes de armazenamento que apontam para um ou outro. Por exemplo, você pode configurar uma `san-dev` classe que usa o `ontap-san` driver e uma `san-default` classe que usa a `ontap-san-economy` mesma.

Todos os seus nós de trabalho do Kubernetes precisam ter as ferramentas iSCSI apropriadas instaladas. "aqui"Consulte para obter mais detalhes.

#### Autenticação

O Astra Trident oferece dois modos de autenticação no back-end do ONTAP.

- Baseado em credenciais: O nome de usuário e senha para um usuário do ONTAP com as permissões necessárias. Recomenda-se a utilização de uma função de início de sessão de segurança predefinida, como `admin` ou `vsadmin` para garantir a máxima compatibilidade com as versões do ONTAP.
- Baseado em certificado: O Astra Trident também pode se comunicar com um cluster ONTAP usando um certificado instalado no back-end. Aqui, a definição de back-end deve conter valores codificados em Base64 do certificado de cliente, chave e certificado de CA confiável, se usado (recomendado).

Os usuários também podem optar por atualizar os backends existentes, optando por mover-se de credenciais

para baseadas em certificados e vice-versa. Se **as credenciais e os certificados forem fornecidos**, o Astra Trident usará os certificados por padrão ao emitir um aviso para remover as credenciais da definição de back-end.

#### Ative a autenticação baseada em credenciais

O Astra Trident requer as credenciais para um administrador com escopo SVM/cluster para se comunicar com o back-end do ONTAP. Recomenda-se a utilização de funções padrão predefinidas, como `admin` ou `vsadmin`. Isso garante compatibilidade direta com futuras versões do ONTAP que podem expor APIs de recursos a serem usadas por futuras versões do Astra Trident. Uma função de login de segurança personalizada pode ser criada e usada com o Astra Trident, mas não é recomendada.

Uma definição de backend de exemplo será assim:

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

Tenha em mente que a definição de back-end é o único lugar onde as credenciais são armazenadas em texto simples. Depois que o back-end é criado, os nomes de usuário/senhas são codificados com Base64 e armazenados como segredos do Kubernetes. A criação/updation de um backend é a única etapa que requer conhecimento das credenciais. Como tal, é uma operação somente de administrador, a ser realizada pelo administrador do Kubernetes/storage.

#### Ativar autenticação baseada em certificado

Backends novos e existentes podem usar um certificado e se comunicar com o back-end do ONTAP. Três parâmetros são necessários na definição de backend.

- ClientCertificate: Valor codificado base64 do certificado do cliente.
- ClientPrivateKey: Valor codificado em base64 da chave privada associada.
- TrustedCACertificate: Valor codificado base64 do certificado CA confiável. Se estiver usando uma CA confiável, esse parâmetro deve ser fornecido. Isso pode ser ignorado se nenhuma CA confiável for usada.

Um fluxo de trabalho típico envolve as etapas a seguir.

**Passos**

1. Gerar um certificado e chave de cliente. Ao gerar, defina Nome Comum (CN) para o usuário ONTAP para autenticar como.

```
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. Adicionar certificado de CA confiável ao cluster do ONTAP. Isso pode já ser Tratado pelo administrador do armazenamento. Ignore se nenhuma CA confiável for usada.

```
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. Instale o certificado e a chave do cliente (a partir do passo 1) no cluster do ONTAP.

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

4. Confirme se a função de login de segurança do ONTAP suporta `cert` o método de autenticação.

```
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. Teste a autenticação usando certificado gerado. Substitua o ONTAP Management LIF> e o <vserver name> por IP de LIF de gerenciamento e nome da 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. Codificar certificado, chave e certificado CA confiável com Base64.

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

7. Crie backend usando os valores obtidos na etapa anterior.

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

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

**Atualizar métodos de autenticação ou girar credenciais**

Você pode atualizar um back-end existente para fazer uso de um método de autenticação diferente ou para girar suas credenciais. Isso funciona de ambas as maneiras: Backends que fazem uso de nome de usuário / senha podem ser atualizados para usar certificados; backends que utilizam certificados podem ser atualizados para nome de usuário / senha com base. Para fazer isso, use um arquivo atualizado `backend.json` contendo os parâmetros necessários para executar `tridentctl backend update`o .

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

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

Ao girar senhas, o administrador de armazenamento deve primeiro atualizar a senha do usuário no ONTAP. Isso é seguido por uma atualização de back-end. Ao girar certificados, vários certificados podem ser adicionados ao usuário. O back-end é então atualizado para usar o novo certificado, seguindo o qual o certificado antigo pode ser excluído do cluster do ONTAP.

A atualização de um back-end não interrompe o acesso a volumes que já foram criados, nem afeta as conexões de volume feitas depois. Uma atualização de back-end bem-sucedida indica que o Astra Trident pode se comunicar com o back-end do ONTAP e lidar com operações de volume futuras.

#### Especifique grupos

O Astra Trident usa os grupos para controlar o acesso aos volumes (LUNs) provisionados. Os administradores têm duas opções quando se trata de especificar grupos para backends:

- O Astra Trident pode criar e gerenciar automaticamente um grupo por back-end. Se `igroupName` não estiver incluído na definição de back-end, o Astra Trident criará um grupo nomeado `trident-<backend-UUID>` no SVM. Isso garantirá que cada back-end tenha um iggroup dedicado e tratará da adição/exclusão automatizada de IQNs do nó Kubernetes.
- Alternativamente, os grupos pré-criados também podem ser fornecidos em uma definição de back-end. Isso pode ser feito usando o `igroupName` parâmetro config. O Astra Trident adicionará/excluirá IQNs de

nós do Kubernetes ao grupo pré-existente.

Para backends que `igroupName` tenham definido, o `igroupName` pode ser excluído com um `tridentctl backend update` para ter os grupos de auto-manipulação Astra Trident. Isso não interromperá o acesso a volumes que já estão anexados a cargas de trabalho. Conexões futuras serão tratadas usando o igroup Astra Trident criado.

> Dedicar um grupo para cada instância única do Astra Trident é uma prática recomendada que é benéfica para o administrador do Kubernetes, bem como para o administrador de storage. O CSI Trident automatiza a adição e remoção de IQNs de nó de cluster ao igrupo, simplificando muito seu gerenciamento. Ao usar o mesmo SVM em ambientes Kubernetes (e instalações Astra Trident), o uso de um grupo dedicado garante que as alterações feitas em um cluster do Kubernetes não influenciem os grupos associados a outro. Além disso, também é importante garantir que cada nó no cluster do Kubernetes tenha uma IQN exclusiva. Como mencionado acima, o Astra Trident lida automaticamente com a adição e remoção de IQNs. A reutilização de IQNs entre hosts pode levar a cenários indesejáveis nos quais os hosts se confundem uns com os outros e o acesso a LUNs é negado.

Se o Astra Trident estiver configurado para funcionar como um supervisor do CSI, os IQNs do nó do Kubernetes serão automaticamente adicionados/removidos do grupo. Quando os nós são adicionados a um cluster Kubernetes, `trident-csi` o DaemonSet implanta um pod (`trident-csi-xxxxx`) nos nós recém-adicionados e Registra os novos nós aos quais pode anexar volumes. Os IQNs de nó também são adicionados ao igroup do back-end. Um conjunto semelhante de etapas manipula a remoção de IQNs quando os nós são cordonados, drenados e excluídos do Kubernetes.

Se o Astra Trident não for executado como um supervisor de CSI, o grupo deve ser atualizado manualmente para conter os IQNs iSCSI de cada nó de trabalho no cluster do Kubernetes. As IQNs de nós que ingressam no cluster do Kubernetes precisarão ser adicionadas ao grupo. Da mesma forma, as IQNs de nós removidos do cluster do Kubernetes devem ser removidas do grupo.

#### Autentique conexões com CHAP bidirecional

O Astra Trident pode autenticar sessões iSCSI com CHAP bidirecional para os `ontap-san` drivers e `ontap-san-economy`. Isso requer a ativação da `useCHAP` opção na definição de backend. Quando definido como `true`, o Astra Trident configura a segurança do iniciador padrão do SVM para CHAP bidirecional e define o nome de usuário e os segredos do arquivo de back-end. O NetApp recomenda o uso de CHAP bidirecional para autenticar conexões. Veja a seguinte configuração de exemplo:

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

O `useCHAP` parâmetro é uma opção booleana que pode ser configurada apenas uma vez. Ele é definido como false por padrão. Depois de configurá-lo como verdadeiro, você não pode configurá-lo como falso.

Além `useCHAP=true` do , os `chapInitiatorSecret` campos , `chapTargetInitiatorSecret`, `chapTargetUsername`, e `chapUsername` devem ser incluídos na definição de back-end. Os segredos podem ser alterados depois que um backend é criado executando `tridentctl update`.

#### Como funciona

Ao definir `useCHAP` como verdadeiro, o administrador de storage instrui o Astra Trident a configurar o CHAP no back-end de storage. Isso inclui o seguinte:

- Configuração do CHAP no SVM:
  - Se o tipo de segurança do iniciador padrão da SVM for nenhum (definido por padrão) **e** não houver LUNs pré-existentes no volume, o Astra Trident definirá o tipo de segurança padrão `CHAP` e continuará configurando o iniciador CHAP e o nome de usuário e os segredos de destino.
  - Se o SVM contiver LUNs, o Astra Trident não ativará o CHAP no SVM. Isso garante que o acesso a LUNs que já estão presentes no SVM não seja restrito.
- Configurando o iniciador CHAP e o nome de usuário e os segredos de destino; essas opções devem ser especificadas na configuração de back-end (como mostrado acima).
- Gerenciando a adição de iniciadores ao `igroupName` dado no back-end. Se não for especificado, o padrão é `trident`.

Depois que o back-end é criado, o Astra Trident cria um CRD correspondente `tridentbackend` e armazena os segredos e nomes de usuário do CHAP como segredos do Kubernetes. Todos os PVS criados pelo Astra Trident neste back-end serão montados e anexados através do CHAP.

#### Gire credenciais e atualize os backends

Você pode atualizar as credenciais CHAP atualizando os parâmetros CHAP no `backend.json` arquivo. Isso

exigirá a atualização dos segredos CHAP e o uso do `tridentctl update` comando para refletir essas alterações.

> Ao atualizar os segredos CHAP para um backend, você deve usar `tridentctl` para atualizar o backend. Não atualize as credenciais no cluster de storage por meio da IU da CLI/ONTAP, pois o Astra Trident não conseguirá aceitar essas alterações.

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

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

As conexões existentes não serão afetadas. Elas continuarão ativas se as credenciais forem atualizadas pelo Astra Trident no SVM. As novas conexões usarão as credenciais atualizadas e as conexões existentes continuam ativas. Desconetar e reconetar PVS antigos resultará em eles usando as credenciais atualizadas.

### Opções de configuração e exemplos

Saiba mais sobre como criar e usar drivers SAN ONTAP com sua instalação do Astra Trident. Esta seção fornece exemplos de configuração de back-end e detalhes sobre como mapear backends para StorageClasses.

#### Opções de configuração de back-end

Consulte a tabela a seguir para obter as opções de configuração de back-end:

| Parâmetro | Descrição | Padrão |
|---|---|---|
| `version` | | Sempre 1 |
| `storageDriverName` | Nome do controlador de armazenamento | "ONTAP-nas", "ONTAP-nas-economy", "ONTAP-nas-FlexGroup", "ONTAP-san", "ONTAP-san-economy" |
| `backendName` | Nome personalizado ou back-end de storage | Nome do driver |
| `managementLIF` | Endereço IP de um cluster ou LIF de gerenciamento de SVM | "10,0.0,1", "[2001:1234:abcd::fefe]" |
| `dataLIF` | Endereço IP do protocolo LIF. Use suportes quadrados para IPv6. Não pode ser atualizado depois de configurá-lo | Derivado do SVM, a menos que especificado |
| `useCHAP` | Usar CHAP para autenticar iSCSI para drivers SAN ONTAP [Boolean] | falso |
| `chapInitiatorSecret` | Segredo do iniciador CHAP. Necessário se `useCHAP=true` | "" |
| `labels` | Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes | "" |
| `chapTargetInitiatorSecret` | Segredo do iniciador de destino CHAP. Necessário se `useCHAP=true` | "" |
| `chapUsername` | Nome de utilizador de entrada. Necessário se `useCHAP=true` | "" |
| `chapTargetUsername` | Nome de utilizador alvo. Necessário se `useCHAP=true` | "" |
| `clientCertificate` | Valor codificado em base64 do certificado do cliente. Usado para autenticação baseada em certificado | "" |
| `clientPrivateKey` | Valor codificado em base64 da chave privada do cliente. Usado para autenticação baseada em certificado | "" |
| `trustedCACertificate` | Valor codificado em base64 do certificado CA confiável. Opcional. Usado para autenticação baseada em certificado | "" |

| Parâmetro | Descrição | Padrão |
|---|---|---|
| `username` | Nome de usuário para se conetar ao cluster/SVM. Usado para autenticação baseada em credenciais | "" |
| `password` | Senha para se conectar ao cluster/SVM. Usado para autenticação baseada em credenciais | "" |
| `svm` | Máquina virtual de armazenamento para usar | Derivado se uma SVM `managementLIF` for especificada |
| `igroupName` | Nome do grupo para volumes SAN a serem usados | "Trident-<backend-UUID>" |
| `storagePrefix` | Prefixo usado ao provisionar novos volumes na SVM. Não pode ser atualizado depois de configurá-lo | "Trident" |
| `limitAggregateUsage` | Falha no provisionamento se o uso estiver acima dessa porcentagem. **Não se aplica ao Amazon FSX for ONTAP** | "" (não aplicado por padrão) |
| `limitVolumeSize` | Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor. | "" (não aplicado por padrão) |
| `lunsPerFlexvol` | Máximo de LUNs por FlexVol, tem de estar no intervalo [50, 200] | "100" |
| `debugTraceFlags` | Debug flags para usar ao solucionar problemas. Por exemplo, "api":false, "método":true" | nulo |
| `useREST` | Parâmetro booleano para usar APIs REST do ONTAP. **Pré-visualização técnica** | falso |

`useREST` é fornecido como uma **prévia técnica** recomendada para ambientes de teste e não para cargas de trabalho de produção. Quando definido como `true`, o Astra Trident usará as APIS REST do ONTAP para se comunicar com o back-end. Esse recurso requer o ONTAP 9.9 e posterior. Além disso, a função de login do ONTAP usada deve ter acesso ao `ontap` aplicativo. Isso é satisfeito com as funções e `cluster-admin` predefinidas `vsadmin`.

Para se comunicar com o cluster ONTAP, você deve fornecer os parâmetros de autenticação. Esse pode ser o nome de usuário/senha para um login de segurança ou um certificado instalado.

Se você estiver usando um back-end do Amazon FSX for NetApp ONTAP, não especifique o `limitAggregateUsage` parâmetro. `fsxadmin`As funções e `vsadmin` fornecidas pelo Amazon FSX para NetApp ONTAP não contêm as permissões de acesso necessárias para recuperar o uso agregado e limitá-lo por meio do Astra Trident.

Não use `debugTraceFlags` a menos que você esteja solucionando problemas e exija um despejo de log detalhado.

Para os `ontap-san` drivers, o padrão é usar todos os IPs de LIF de dados da SVM e usar multipath iSCSI. Especificar um endereço IP para o dataLIF para os `ontap-san` drivers obriga-os a desabilitar o multipath e usar apenas o endereço especificado.

Ao criar um backend, lembre-se disso `dataLIF` e `storagePrefix` não pode ser modificado após a criação. Para atualizar esses parâmetros, você precisará criar um novo backend.

`igroupName` Pode ser definido como um grupo que já está criado no cluster ONTAP. Se não for especificado, o Astra Trident cria automaticamente um grupo chamado Trident-<backend-UUID>. Se estiver fornecendo um nome de grupo predefinido, o NetApp recomenda o uso de um grupo por cluster do Kubernetes, se o SVM for compartilhado entre ambientes. Isso é necessário para que o Astra Trident mantenha automaticamente adições/exclusões ao IQN.

Os backends também podem ter grupos atualizados após a criação:

- O igroup Name pode ser atualizado para apontar para um novo igroup que é criado e gerenciado no SVM fora do Astra Trident.
- O igroupName pode ser omitido. Nesse caso, o Astra Trident criará e gerenciará um grupo Trident-<backend-UUID> automaticamente.

Em ambos os casos, os anexos de volume continuarão a ser acessíveis. Futuros anexos de volume usarão o igroup atualizado. Esta atualização não interrompe o acesso aos volumes presentes no back-end.

Um nome de domínio totalmente qualificado (FQDN) pode ser especificado para a `managementLIF` opção.

```
`managementLIF` Para todos os drivers ONTAP também pode ser definido como
endereços IPv6. Certifique-se de que instala o Trident com o `--use-ipv6`
sinalizador. Deve-se ter cuidado para definir `managementLIF` o endereço
IPv6 entre parênteses retos.
```

Ao usar endereços IPv6, certifique-se de `managementLIF` que e `dataLIF` (se incluídos na definição do backend) estejam definidos entre colchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Se `dataLIF` não for fornecido, o Astra Trident irá buscar os LIFs de dados do IPv6 do SVM.

Para habilitar os drivers ONTAP-san para usar o CHAP, defina o `useCHAP` parâmetro como `true` em sua definição de back-end. Em seguida, o Astra Trident configurará e usará CHAP bidirecional como a autenticação padrão para a SVM fornecida no back-end. "aqui"Consulte para saber como funciona.

Para `ontap-san-economy` o driver, a `limitVolumeSize` opção também restringirá o tamanho máximo dos volumes que gerencia para qtrees e LUNs.

O Astra Trident define rótulos de provisionamento no campo "Comentários" de todos os volumes criados usando `ontap-san` o driver. Para cada volume criado, o campo "Comentários" no FlexVol será preenchido com todas as etiquetas presentes no pool de armazenamento em que ele é colocado. Os administradores de armazenamento podem definir rótulos por pool de armazenamento e agrupar todos os volumes criados em um pool de armazenamento. Isso fornece uma maneira conveniente de diferenciar volumes com base em um conjunto de rótulos personalizáveis que são fornecidos na configuração de back-end.

**Opções de configuração de back-end para volumes de provisionamento**

Você pode controlar como cada volume é provisionado por padrão usando essas opções em uma seção especial da configuração. Para obter um exemplo, consulte os exemplos de configuração abaixo.

| Parâmetro | Descrição | Padrão |
|---|---|---|
| `spaceAllocation` | Alocação de espaço para LUNs | "verdadeiro" |
| `spaceReserve` | Modo de reserva de espaço; "nenhum" (fino) ou "volume" (grosso) | "nenhum" |
| `snapshotPolicy` | Política de instantâneos a utilizar | "nenhum" |
| `qosPolicy` | Grupo de políticas de QoS a atribuir aos volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento/backend | "" |
| `adaptiveQosPolicy` | Grupo de políticas de QoS adaptável a atribuir para volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento/backend | "" |
| `snapshotReserve` | Porcentagem de volume reservado para snapshots "0" | Se `snapshotPolicy` é "nenhum", então "" |
| `splitOnClone` | Divida um clone de seu pai na criação | "falso" |
| `splitOnClone` | Divida um clone de seu pai na criação | "falso" |
| `encryption` | Ative a criptografia de volume do NetApp | "falso" |
| `securityStyle` | Estilo de segurança para novos volumes | "unix" |
| `tieringPolicy` | Política de disposição em camadas para usar "nenhuma" | "Somente snapshot" para configuração pré-ONTAP 9.5 SVM-DR |

O uso de grupos de política de QoS com o Astra Trident requer o ONTAP 9.8 ou posterior. Recomenda-se usar um grupo de políticas QoS não compartilhado e garantir que o grupo de políticas seja aplicado individualmente a cada componente. Um grupo de política de QoS compartilhado aplicará o limite máximo da taxa de transferência total de todos os workloads.

Aqui está um exemplo com padrões definidos:

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

Para todos os volumes criados com `ontap-san` o driver, o Astra Trident adiciona uma capacidade extra de 10% ao FlexVol para acomodar os metadados do LUN. O LUN será provisionado com o tamanho exato que o usuário solicita no PVC. O Astra Trident adiciona 10% ao FlexVol (mostra como tamanho disponível no ONTAP). Os usuários agora terão a capacidade utilizável que solicitaram. Essa alteração também impede que LUNs fiquem somente leitura, a menos que o espaço disponível seja totalmente utilizado. Isto não se aplica à ONTAP-san-economia.

Para backends que definem `snapshotReserve`o , o Astra Trident calcula o tamanho dos volumes da seguinte forma:

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

O 1,1 é o 10% adicional que o Astra Trident adiciona ao FlexVol para acomodar os metadados do LUN. Para `snapshotReserve` 5%, e o pedido de PVC é de 5GiB, o tamanho total do volume é de 5,79GiB e o tamanho disponível é de 5,5GiB. O `volume show` comando deve mostrar resultados semelhantes a este exemplo:


Atualmente, o redimensionamento é a única maneira de usar o novo cálculo para um volume existente.

#### Exemplos mínimos de configuração

Os exemplos a seguir mostram configurações básicas que deixam a maioria dos parâmetros padrão. Esta é a maneira mais fácil de definir um backend.

> Se você estiver usando o Amazon FSX no NetApp ONTAP com Astra Trident, a recomendação é especificar nomes DNS para LIFs em vez de endereços IP.

##### `ontap-san` driver com autenticação baseada em certificado

Este é um exemplo de configuração de back-end mínimo. `clientCertificate`, `clientPrivateKey` E `trustedCACertificate` (opcional, se estiver usando CA confiável) são preenchidos `backend.json` e recebem os valores codificados em base64 do certificado do cliente, da chave privada e do certificado de CA confiável, respetivamente.

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

##### `ontap-san` Driver com CHAP bidirecional

Este é um exemplo de configuração de back-end mínimo. Essa configuração básica cria um `ontap-san` back-end com `useCHAP` definido como `true`.

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

`ontap-san-economy` **condutor**

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

#### Exemplos de backends com pools de armazenamento virtual

No arquivo de definição de back-end de exemplo mostrado abaixo, padrões específicos são definidos para todos os pools de armazenamento, como `spaceReserve` em nenhum, `spaceAllocation` em falso e `encryption` em falso. Os pools de armazenamento virtual são definidos na seção armazenamento.

Neste exemplo, alguns dos conjuntos de armazenamento definem os seus próprios `spaceReserve` `spaceAllocation` valores , e `encryption` , e alguns conjuntos substituem os valores predefinidos acima.

```
```

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

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

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

Aqui está um exemplo iSCSI para `ontap-san-economy` o driver:

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

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

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

#### Mapeie os backends para StorageClasses

As seguintes definições do StorageClass referem-se aos pools de armazenamento virtual acima. Usando o `parameters.selector` campo, cada StorageClass chama qual(s) pool(s) virtual(s) pode(m) ser(ão) usado(s) para hospedar um volume. O volume terá os aspetos definidos no pool virtual escolhido.

- O primeiro StorageClass ) (`protection-gold`será mapeado para o primeiro e segundo pool de armazenamento virtual `ontap-nas-flexgroup` no back-end e o primeiro pool de armazenamento virtual `ontap-san` no back-end. Estas são as únicas piscinas que oferecem proteção de nível de ouro.
- O segundo StorageClass ) (`protection-not-gold`será mapeado para o terceiro, quarto pool de armazenamento virtual no `ontap-nas-flexgroup` back-end e o segundo, terceiro pool de armazenamento virtual `ontap-san` no back-end. Estas são as únicas piscinas que oferecem um nível de proteção diferente do ouro.
- O terceiro StorageClass ) (`app-mysqldb`será mapeado para o quarto pool de armazenamento virtual no `ontap-nas` back-end e o terceiro pool de armazenamento virtual `ontap-san-economy` no back-end. Estes são os únicos pools que oferecem configuração de pool de armazenamento para o aplicativo do tipo mysqldb.
- O quarto StorageClass ) (`protection-silver-creditpoints-20k`será mapeado para o terceiro pool de armazenamento virtual no `ontap-nas-flexgroup` back-end e o segundo pool de armazenamento virtual `ontap-san` no back-end. Estas são as únicas piscinas que oferecem proteção de nível dourado em 20000 pontos de crédito.
- O quinto StorageClass ) (`creditpoints-5k`será mapeado para o segundo pool de armazenamento virtual `ontap-nas-economy` no back-end e o terceiro pool de armazenamento virtual `ontap-san` no back-end. Estas são as únicas ofertas de pool em 5000 pontos de crédito.

O Astra Trident decidirá qual pool de storage virtual está selecionado e garantirá que o requisito de storage seja atendido.

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

# Configurar um back-end com drivers nas ONTAP

Saiba mais sobre como configurar um back-end ONTAP com drivers nas ONTAP e Cloud Volumes ONTAP.

- "Preparação"
- "Configuração e exemplos"

### Permissões do usuário

O Astra Trident espera ser executado como administrador da ONTAP ou SVM, normalmente usando o `admin` usuário do cluster ou um `vsadmin` usuário SVM, ou um usuário com um nome diferente que tenha a mesma função. Para implantações do Amazon FSX for NetApp ONTAP, o Astra Trident espera ser executado como administrador do ONTAP ou SVM, usando o usuário do cluster `fsxadmin` ou um `vsadmin` usuário SVM, ou um usuário com um nome diferente que tenha a mesma função. O `fsxadmin` usuário é um substituto limitado para o usuário administrador do cluster.

> Se você usar o `limitAggregateUsage` parâmetro, as permissões de administrador do cluster serão necessárias. Ao usar o Amazon FSX for NetApp ONTAP com Astra Trident, o `limitAggregateUsage` parâmetro não funcionará com as `vsadmin` contas de usuário e `fsxadmin`. A operação de configuração falhará se você especificar este parâmetro.

Embora seja possível criar uma função mais restritiva no ONTAP que um driver Trident pode usar, não recomendamos. A maioria das novas versões do Trident chamarão APIs adicionais que teriam que ser contabilizadas, tornando as atualizações difíceis e suscetíveis a erros.

### Preparação

Saiba mais sobre como se preparar para configurar um back-end ONTAP com drivers NAS ONTAP. Para todos os back-ends ONTAP, o Astra Trident requer pelo menos um agregado atribuído ao SVM.

Para todos os back-ends ONTAP, o Astra Trident requer pelo menos um agregado atribuído ao SVM.

Lembre-se de que você também pode executar mais de um driver e criar classes de armazenamento que apontam para um ou outro. Por exemplo, você pode configurar uma classe Gold que usa o `ontap-nas` driver e uma classe Bronze que usa o `ontap-nas-economy` um.

Todos os seus nós de trabalho do Kubernetes precisam ter as ferramentas NFS apropriadas instaladas. "aqui"Consulte para obter mais detalhes.

#### Autenticação

O Astra Trident oferece dois modos de autenticação no back-end do ONTAP.

- Baseado em credenciais: O nome de usuário e senha para um usuário do ONTAP com as permissões necessárias. Recomenda-se a utilização de uma função de início de sessão de segurança predefinida, como `admin` ou `vsadmin` para garantir a máxima compatibilidade com as versões do ONTAP.
- Baseado em certificado: O Astra Trident também pode se comunicar com um cluster ONTAP usando um certificado instalado no back-end. Aqui, a definição de back-end deve conter valores codificados em Base64 do certificado de cliente, chave e certificado de CA confiável, se usado (recomendado).

Os usuários também podem optar por atualizar os backends existentes, optando por mover-se de credenciais para baseadas em certificados e vice-versa. Se **as credenciais e os certificados forem fornecidos**, o Astra Trident usará os certificados por padrão ao emitir um aviso para remover as credenciais da definição de back-

end.

### Ative a autenticação baseada em credenciais

O Astra Trident requer as credenciais para um administrador com escopo SVM/cluster para se comunicar com o back-end do ONTAP. Recomenda-se a utilização de funções padrão predefinidas, como `admin` ou `vsadmin`. Isso garante compatibilidade direta com futuras versões do ONTAP que podem expor APIs de recursos a serem usadas por futuras versões do Astra Trident. Uma função de login de segurança personalizada pode ser criada e usada com o Astra Trident, mas não é recomendada.

Uma definição de backend de exemplo será assim:

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

Tenha em mente que a definição de back-end é o único lugar onde as credenciais são armazenadas em texto simples. Depois que o back-end é criado, os nomes de usuário/senhas são codificados com Base64 e armazenados como segredos do Kubernetes. A criação/updation de um backend é a única etapa que requer conhecimento das credenciais. Como tal, é uma operação somente de administrador, a ser realizada pelo administrador do Kubernetes/storage.

### Ativar autenticação baseada em certificado

Backends novos e existentes podem usar um certificado e se comunicar com o back-end do ONTAP. Três parâmetros são necessários na definição de backend.

- ClientCertificate: Valor codificado base64 do certificado do cliente.
- ClientPrivateKey: Valor codificado em base64 da chave privada associada.
- TrustedCACertificate: Valor codificado base64 do certificado CA confiável. Se estiver usando uma CA confiável, esse parâmetro deve ser fornecido. Isso pode ser ignorado se nenhuma CA confiável for usada.

Um fluxo de trabalho típico envolve as etapas a seguir.

#### Passos

1. Gerar um certificado e chave de cliente. Ao gerar, defina Nome Comum (CN) para o usuário ONTAP para autenticar como.

```
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. Adicionar certificado de CA confiável ao cluster do ONTAP. Isso pode já ser Tratado pelo administrador do armazenamento. Ignore se nenhuma CA confiável for usada.

```
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. Instale o certificado e a chave do cliente (a partir do passo 1) no cluster do ONTAP.

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

4. Confirme se a função de login de segurança do ONTAP suporta `cert` o método de autenticação.

```
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. Teste a autenticação usando certificado gerado. Substitua o ONTAP Management LIF> e o <vserver name> por IP de LIF de gerenciamento e nome da SVM. Você deve garantir que o LIF tenha sua política de serviço definida como `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. Codificar certificado, chave e certificado CA confiável com Base64.

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

7. Crie backend usando os valores obtidos na etapa anterior.

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

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

**Atualizar métodos de autenticação ou girar credenciais**

Você pode atualizar um back-end existente para fazer uso de um método de autenticação diferente ou para girar suas credenciais. Isso funciona de ambas as maneiras: Backends que fazem uso de nome de usuário / senha podem ser atualizados para usar certificados; backends que utilizam certificados podem ser atualizados para nome de usuário / senha com base. Para fazer isso, use um arquivo atualizado `backend.json` contendo os parâmetros necessários para executar `tridentctl backend update`o .

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

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

Ao girar senhas, o administrador de armazenamento deve primeiro atualizar a senha do usuário no ONTAP. Isso é seguido por uma atualização de back-end. Ao girar certificados, vários certificados podem ser adicionados ao usuário. O back-end é então atualizado para usar o novo certificado, seguindo o qual o certificado antigo pode ser excluído do cluster do ONTAP.

A atualização de um back-end não interrompe o acesso a volumes que já foram criados, nem afeta as conexões de volume feitas depois. Uma atualização de back-end bem-sucedida indica que o Astra Trident pode se comunicar com o back-end do ONTAP e lidar com operações de volume futuras.

#### Gerenciar políticas de exportação de NFS

O Astra Trident usa políticas de exportação de NFS para controlar o acesso aos volumes provisionados.

O Astra Trident oferece duas opções ao trabalhar com políticas de exportação:

- O Astra Trident pode gerenciar dinamicamente a própria política de exportação; nesse modo de operação, o administrador de armazenamento especifica uma lista de blocos CIDR que representam endereços IP admissíveis. O Astra Trident adiciona IPs de nós que se enquadram nesses intervalos à política de exportação automaticamente. Como alternativa, quando nenhum CIDR é especificado, qualquer IP unicast de escopo global encontrado nos nós será adicionado à política de exportação.

- Os administradores de storage podem criar uma política de exportação e adicionar regras manualmente. O Astra Trident usa a política de exportação padrão, a menos que um nome de política de exportação diferente seja especificado na configuração.

**Gerencie dinamicamente políticas de exportação**

A versão 20,04 do CSI Trident oferece a capacidade de gerenciar dinamicamente políticas de exportação para backends ONTAP. Isso fornece ao administrador de armazenamento a capacidade de especificar um espaço de endereço permitido para IPs de nó de trabalho, em vez de definir regras explícitas manualmente. Ele simplifica muito o gerenciamento de políticas de exportação. As modificações na política de exportação não exigem mais intervenção manual no cluster de storage. Além disso, isso ajuda a restringir o acesso ao cluster de armazenamento somente aos nós de trabalho que têm IPs no intervalo especificado, suportando um gerenciamento automatizado e refinado.

O gerenciamento dinâmico das políticas de exportação está disponível apenas para o CSI Trident. É importante garantir que os nós de trabalho não estejam sendo repartidos.

**Exemplo**

Há duas opções de configuração que devem ser usadas. Aqui está um exemplo de definição de back-end:

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

Ao usar esse recurso, você deve garantir que a junção raiz do SVM tenha uma política de exportação pré-ajustada com uma regra de exportação que permita o bloco CIDR do nó (como a política de exportação padrão). Siga sempre as práticas recomendadas pela NetApp para dedicar um SVM ao Astra Trident.

Aqui está uma explicação de como esse recurso funciona usando o exemplo acima:

- `autoExportPolicy` está definido como `true`. Isso indica que o Astra Trident criará uma política de exportação para `svm1` o SVM e tratará da adição e exclusão de regras usando `autoExportCIDRs` blocos de endereço. Por exemplo, um back-end com UUID 403b5326-8482-40dB-96d0-d83fb3f4daec e `autoExportPolicy` definido como `true` cria uma política de exportação nomeada `trident-403b5326-8482-40db-96d0-d83fb3f4daec` no SVM.
- `autoExportCIDRs` contém uma lista de blocos de endereços. Este campo é opcional e o padrão é ["0,0.0,0/0", "::/0"]. Se não estiver definido, o Astra Trident adiciona todos os endereços unicast de escopo global encontrados nos nós de trabalho.

Neste exemplo, o `192.168.0.0/24` espaço de endereço é fornecido. Isso indica que os IPs de nós do Kubernetes que se enquadram nesse intervalo de endereços serão adicionados à política de exportação criada pelo Astra Trident. Quando o Astra Trident Registra um nó em que ele é executado, ele recupera os endereços IP do nó e os verifica em relação aos blocos de endereço fornecidos no `autoExportCIDRs`. depois de filtrar os IPs, o Astra Trident cria regras de política de exportação para os IPs de cliente que ele descobre, com uma regra para cada nó que identifica.

Você pode atualizar `autoExportPolicy` e `autoExportCIDRs` para backends depois de criá-los. Você pode anexar novos CIDR para um back-end que é gerenciado automaticamente ou excluir CIDR existentes. Tenha cuidado ao excluir CIDR para garantir que as conexões existentes não sejam descartadas. Você também pode optar por desativar `autoExportPolicy` um back-end e retornar a uma política de exportação criada manualmente. Isso exigirá a configuração do `exportPolicy` parâmetro em sua configuração de backend.

Depois que o Astra Trident criar ou atualizar um back-end, você pode verificar o back-end usando `tridentctl` ou o CRD correspondente `tridentbackend`:

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

Conforme os nós são adicionados a um cluster do Kubernetes e registrados na controladora Astra Trident, as políticas de exportação dos back-ends existentes são atualizadas (desde que elas estejam no intervalo de endereços especificado `autoExportCIDRs` no back-end).

Quando um nó é removido, o Astra Trident verifica todos os back-ends on-line para remover a regra de acesso do nó. Ao remover esse IP de nó das políticas de exportação de backends gerenciados, o Astra Trident impede montagens fraudulentas, a menos que esse IP seja reutilizado por um novo nó no cluster.

Para backends existentes anteriormente, a atualização do back-end com `tridentctl update backend` garantirá que o Astra Trident gerencie as políticas de exportação automaticamente. Isso criará uma nova política de exportação nomeada após o UUID do back-end e os volumes presentes no back-end usarão a política de exportação recém-criada quando forem montados novamente.

A exclusão de um back-end com políticas de exportação gerenciadas automaticamente excluirá a política de exportação criada dinamicamente. Se o backend for recriado, ele será Tratado como um novo backend e resultará na criação de uma nova política de exportação.

Se o endereço IP de um nó ativo for atualizado, será necessário reiniciar o pod Astra Trident no nó. Em seguida, o Astra Trident atualizará a política de exportação para backends que ele conseguir refletir essa alteração de IP.

### Opções de configuração e exemplos

Saiba mais sobre como criar e usar drivers NAS ONTAP com sua instalação do Astra Trident. Esta seção fornece exemplos de configuração de back-end e detalhes sobre como mapear backends para StorageClasses.

#### Opções de configuração de back-end

Consulte a tabela a seguir para obter as opções de configuração de back-end:

| Parâmetro | Descrição | Padrão |
|---|---|---|
| `version` | | Sempre 1 |
| `storageDriverName` | Nome do controlador de armazenamento | "ONTAP-nas", "ONTAP-nas-economy", "ONTAP-nas-FlexGroup", "ONTAP-san", "ONTAP-san-economy" |
| `backendName` | Nome personalizado ou back-end de storage | Nome do driver |
| `managementLIF` | Endereço IP de um cluster ou LIF de gerenciamento de SVM | "10,0.0,1", "[2001:1234:abcd::fefe]" |
| `dataLIF` | Endereço IP do protocolo LIF. Use suportes quadrados para IPv6. Não pode ser atualizado depois de configurá-lo | Derivado do SVM, a menos que especificado |
| `autoExportPolicy` | Ativar criação e atualização automática de políticas de exportação [Boolean] | falso |
| `autoExportCIDRs` | Lista de CIDR para filtrar IPs de nós do Kubernetes em relação ao `autoExportPolicy` quando o está ativado | ["0,0.0,0/0", "::/0"]» |
| `labels` | Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes | "" |
| `clientCertificate` | Valor codificado em base64 do certificado do cliente. Usado para autenticação baseada em certificado | "" |

| Parâmetro | Descrição | Padrão |
| --- | --- | --- |
| `clientPrivateKey` | Valor codificado em base64 da chave privada do cliente. Usado para autenticação baseada em certificado | "" |
| `trustedCACertificate` | Valor codificado em base64 do certificado CA confiável. Opcional. Usado para autenticação baseada em certificado | "" |
| `username` | Nome de usuário para se conetar ao cluster/SVM. Usado para autenticação baseada em credenciais | |
| `password` | Senha para se conectar ao cluster/SVM. Usado para autenticação baseada em credenciais | |
| `svm` | Máquina virtual de armazenamento para usar | Derivado se uma SVM `managementLIF` for especificada |
| `igroupName` | Nome do grupo para volumes SAN a serem usados | "Trident-<backend-UUID>" |
| `storagePrefix` | Prefixo usado ao provisionar novos volumes na SVM. Não pode ser atualizado depois de configurá-lo | "Trident" |
| `limitAggregateUsage` | Falha no provisionamento se o uso estiver acima dessa porcentagem. **Não se aplica ao Amazon FSX for ONTAP** | "" (não aplicado por padrão) |
| `limitVolumeSize` | Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor. | "" (não aplicado por padrão) |
| `lunsPerFlexvol` | Máximo de LUNs por FlexVol, tem de estar no intervalo [50, 200] | "100" |
| `debugTraceFlags` | Debug flags para usar ao solucionar problemas. Por exemplo, "api":false, "método":true" | nulo |
| `nfsMountOptions` | Lista separada por vírgulas de opções de montagem NFS | "" |
| `qtreesPerFlexvol` | Qtrees máximos por FlexVol, têm de estar no intervalo [50, 300] | "200" |
| `useREST` | Parâmetro booleano para usar APIs REST do ONTAP. **Pré-visualização técnica** | falso |

`useREST` é fornecido como uma **prévia técnica** recomendada para ambientes de teste e não para cargas de trabalho de produção. Quando definido como `true`, o Astra Trident usará as APIS REST do ONTAP para se comunicar com o back-end. Esse recurso requer o ONTAP 9.9 e posterior. Além disso, a função de login do ONTAP usada deve ter acesso ao `ontap` aplicativo. Isso é satisfeito com as funções e `cluster-admin` predefinidas `vsadmin`.

Para se comunicar com o cluster ONTAP, você deve fornecer os parâmetros de autenticação. Esse pode ser o nome de usuário/senha para um login de segurança ou um certificado instalado.

Se você estiver usando um back-end do Amazon FSX for NetApp ONTAP, não especifique o `limitAggregateUsage` parâmetro. `fsxadmin`As funções e `vsadmin` fornecidas pelo Amazon FSX para NetApp ONTAP não contêm as permissões de acesso necessárias para recuperar o uso agregado e limitá-lo por meio do Astra Trident.

Não use `debugTraceFlags` a menos que você esteja solucionando problemas e exija um despejo de log detalhado.

Ao criar um backend, lembre-se de que o `dataLIF` e `storagePrefix` não pode ser modificado após a criação. Para atualizar esses parâmetros, você precisará criar um novo backend.

Um nome de domínio totalmente qualificado (FQDN) pode ser especificado para a `managementLIF` opção. Um FQDN também pode ser especificado para a `dataLIF` opção, caso em que o FQDN será usado para as operações de montagem NFS. Dessa forma, você pode criar um DNS de round-robin para balanceamento de carga em vários LIFs de dados.

```
`managementLIF` Para todos os drivers ONTAP também pode ser definido como endereços IPv6. Certifique-se de instalar o Astra Trident com o `--use-ipv6` sinalizador. Deve-se ter cuidado para definir o `managementLIF` endereço IPv6 entre parênteses retos.
```

Ao usar endereços IPv6, certifique-se de `managementLIF` que e `dataLIF` (se incluídos na definição do backend) estejam definidos entre colchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Se `dataLIF` não for fornecido, o Astra Trident irá buscar os LIFs de dados do IPv6 do SVM.

Usando as `autoExportPolicy` opções e `autoExportCIDRs`, o CSI Trident pode gerenciar políticas de exportação automaticamente. Isso é compatível com todos os drivers ONTAP-nas-*.

Para o `ontap-nas-economy` driver, a `limitVolumeSize` opção também restringirá o tamanho máximo dos volumes que gerencia para qtrees e LUNs, e a `qtreesPerFlexvol` opção permite personalizar o número máximo de qtrees por FlexVol.

O `nfsMountOptions` parâmetro pode ser usado para especificar opções de montagem. As opções de montagem para volumes persistentes do Kubernetes normalmente são especificadas em classes de storage, mas se nenhuma opção de montagem for especificada em uma classe de storage, o Astra Trident voltará a usar as opções de montagem especificadas no arquivo de configuração do back-end de storage. Se nenhuma opção de montagem for especificada na classe de storage ou no arquivo de configuração, o Astra Trident não definirá nenhuma opção de montagem em um volume persistente associado.

O Astra Trident define rótulos de provisionamento no campo "Comentários" de todos os volumes criados usando(`ontap-nas` e(`ontap-nas-flexgroup`. Com base no driver usado, os comentários são definidos no FlexVol (`ontap-nas`) ou no FlexGroup (`ontap-nas-flexgroup`). O Astra Trident copiará todas as etiquetas presentes em um pool de storage para o volume de storage no momento em que ele for provisionado. Os administradores de storage podem definir rótulos por pool de storage e agrupar todos os volumes criados em um pool de storage. Isso fornece uma maneira conveniente de diferenciar volumes com base em um conjunto de rótulos personalizáveis que são fornecidos na configuração de back-end.

**Opções de configuração de back-end para volumes de provisionamento**

Você pode controlar como cada volume é provisionado por padrão usando essas opções em uma seção especial da configuração. Para obter um exemplo, consulte os exemplos de configuração abaixo.

| Parâmetro | Descrição | Padrão |
|---|---|---|
| `spaceAllocation` | Alocação de espaço para LUNs | "verdadeiro" |
| `spaceReserve` | Modo de reserva de espaço; "nenhum" (fino) ou "volume" (grosso) | "nenhum" |
| `snapshotPolicy` | Política de instantâneos a utilizar | "nenhum" |
| `qosPolicy` | Grupo de políticas de QoS a atribuir aos volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento/backend | "" |
| `adaptiveQosPolicy` | Grupo de políticas de QoS adaptável a atribuir para volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento/backend. Não suportado pela ONTAP-nas-Economy. | "" |
| `snapshotReserve` | Porcentagem de volume reservado para snapshots "0" | Se `snapshotPolicy` é "nenhum", então "" |
| `splitOnClone` | Divida um clone de seu pai na criação | "falso" |
| `encryption` | Ative a criptografia de volume do NetApp | "falso" |
| `securityStyle` | Estilo de segurança para novos volumes | "unix" |
| `tieringPolicy` | Política de disposição em camadas para usar "nenhuma" | "Somente snapshot" para configuração pré-ONTAP 9.5 SVM-DR |
| UnixPermissions | Modo para novos volumes | "777" |

| Parâmetro | Descrição | Padrão |
|---|---|---|
| Snapshotdir | Controla a visibilidade `.snapshot` do diretório | "falso" |
| Política de exportação | Política de exportação a utilizar | "padrão" |
| Estilo de segurança | Estilo de segurança para novos volumes | "unix" |

O uso de grupos de política de QoS com o Astra Trident requer o ONTAP 9.8 ou posterior. Recomenda-se usar um grupo de políticas QoS não compartilhado e garantir que o grupo de políticas seja aplicado individualmente a cada componente. Um grupo de política de QoS compartilhado aplicará o limite máximo da taxa de transferência total de todos os workloads.

Aqui está um exemplo com padrões definidos:

```
{
  "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"
  }
}
```

Para `ontap-nas` e `ontap-nas-flexgroups`, o Astra Trident agora usa um novo cálculo para garantir que o FlexVol seja dimensionado corretamente com a porcentagem de snapshotServe e PVC. Quando o usuário solicita um PVC, o Astra Trident cria o FlexVol original com mais espaço usando o novo cálculo. Esse cálculo garante que o usuário receba o espaço gravável que solicitou no PVC, e não menor espaço do que o que solicitou. Antes de v21,07, quando o usuário solicita um PVC (por exemplo, 5GiB), com o snapshotServe a 50 por cento, eles recebem apenas 2,5GiBMB de espaço gravável. Isso ocorre porque o que o usuário solicitou é todo o volume e `snapshotReserve` é uma porcentagem disso. Com o Trident 21,07, o que o usuário solicita é o espaço gravável e o Astra Trident define o `snapshotReserve` número como a porcentagem de todo o volume. Isto não se aplica `ontap-nas-economy` ao . Veja o exemplo a seguir para ver como isso funciona:

O cálculo é o seguinte:

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

Para snapshotServe de 50%, e a solicitação de PVC de 5GiB, o volume total é de 2/.5 10GiB e o tamanho disponível é de 5GiB, o que o usuário solicitou na solicitação de PVC. O `volume show` comando deve mostrar resultados semelhantes a este exemplo:


Os back-ends existentes de instalações anteriores provisionarão volumes conforme explicado acima ao atualizar o Astra Trident. Para volumes que você criou antes da atualização, você deve redimensionar seus volumes para que a alteração seja observada. Por exemplo, um PVC de 2GiB mm com `snapshotReserve=50` anterior resultou em um volume que fornece 1GiB GB de espaço gravável. Redimensionar o volume para 3GiB, por exemplo, fornece ao aplicativo 3GiBMB de espaço gravável em um volume de 6 GiB.

#### Exemplos mínimos de configuração

Os exemplos a seguir mostram configurações básicas que deixam a maioria dos parâmetros padrão. Esta é a maneira mais fácil de definir um backend.

> Se você estiver usando o Amazon FSX no NetApp ONTAP com Trident, a recomendação é especificar nomes DNS para LIFs em vez de endereços IP.

##### `ontap-nas` driver com autenticação baseada em certificado

Este é um exemplo de configuração de back-end mínimo. `clientCertificate`, `clientPrivateKey` E `trustedCACertificate` (opcional, se estiver usando CA confiável) são preenchidos `backend.json` e recebem os valores codificados em base64 do certificado do cliente, da chave privada e do certificado de CA confiável, respetivamente.

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

### `ontap-nas` driver com política de exportação automática

Este exemplo mostra como você pode instruir o Astra Trident a usar políticas de exportação dinâmicas para criar e gerenciar a política de exportação automaticamente. Isso funciona da mesma forma para os `ontap-nas-economy` drivers e `ontap-nas-flexgroup`.

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

### `ontap-nas-flexgroup` condutor

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "labels": {"k8scluster": "test-cluster-east-1b", "backend": "test1-
ontap-cluster"},
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",
}
```

`ontap-nas` **Motorista com IPv6**

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

`ontap-nas-economy` **condutor**

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

#### Exemplos de backends com pools de armazenamento virtual

No arquivo de definição de back-end de exemplo mostrado abaixo, padrões específicos são definidos para todos os pools de armazenamento, como `spaceReserve` em nenhum, `spaceAllocation` em falso e `encryption` em falso. Os pools de armazenamento virtual são definidos na seção armazenamento.

Neste exemplo, alguns dos conjuntos de armazenamento definem os seus próprios `spaceReserve` `spaceAllocation` valores , e `encryption` , e alguns conjuntos substituem os valores predefinidos acima.

`ontap-nas` **condutor**

```
{
    {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "admin",
    "password": "secret",
    "nfsMountOptions": "nfsvers=4",

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

```
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0775"
            }
        },
        {
            "labels":{"app":"mysqldb", "cost":"25"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
        }
    ]
}
```

`ontap-nas-flexgroup` **condutor**

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

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

```
        {
            "labels":{"protection":"gold", "creditpoints":"30000"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"protection":"silver", "creditpoints":"20000"},
            "zone":"us_east_1c",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0775"
            }
        },
        {
            "labels":{"protection":"bronze", "creditpoints":"10000"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
        }
    ]
}
```

`ontap-nas-economy` **condutor**

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

    "defaults": {
          "spaceReserve": "none",
          "encryption": "false"
    },
```

```
    "labels":{"store":"nas_economy_store"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"department":"finance", "creditpoints":"6000"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"department":"legal", "creditpoints":"5000"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"department":"engineering", "creditpoints":"3000"},
            "zone":"us_east_1c",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0775"
            }
        },
        {
            "labels":{"department":"humanresource",
"creditpoints":"2000"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
        }
    ]
}
```

#### Mapeie os backends para StorageClasses

As seguintes definições do StorageClass referem-se aos pools de armazenamento virtual acima. Usando o

`parameters.selector` campo, cada StorageClass chama qual(s) pool(s) virtual(s) pode(m) ser(ão) usado(s) para hospedar um volume. O volume terá os aspetos definidos no pool virtual escolhido.

- O primeiro StorageClass ) (`protection-gold`será mapeado para o primeiro e segundo pool de armazenamento virtual `ontap-nas-flexgroup` no back-end e o primeiro pool de armazenamento virtual `ontap-san` no back-end. Estas são as únicas piscinas que oferecem proteção de nível de ouro.
- O segundo StorageClass ) (`protection-not-gold`será mapeado para o terceiro, quarto pool de armazenamento virtual no `ontap-nas-flexgroup` back-end e o segundo, terceiro pool de armazenamento virtual `ontap-san` no back-end. Estas são as únicas piscinas que oferecem um nível de proteção diferente do ouro.
- O terceiro StorageClass ) (`app-mysqldb`será mapeado para o quarto pool de armazenamento virtual no `ontap-nas` back-end e o terceiro pool de armazenamento virtual `ontap-san-economy` no back-end. Estes são os únicos pools que oferecem configuração de pool de armazenamento para o aplicativo do tipo mysqldb.
- O quarto StorageClass ) (`protection-silver-creditpoints-20k`será mapeado para o terceiro pool de armazenamento virtual no `ontap-nas-flexgroup` back-end e o segundo pool de armazenamento virtual `ontap-san` no back-end. Estas são as únicas piscinas que oferecem proteção de nível dourado em 20000 pontos de crédito.
- O quinto StorageClass ) (`creditpoints-5k`será mapeado para o segundo pool de armazenamento virtual `ontap-nas-economy` no back-end e o terceiro pool de armazenamento virtual `ontap-san` no back-end. Estas são as únicas ofertas de pool em 5000 pontos de crédito.

O Astra Trident decidirá qual pool de storage virtual está selecionado e garantirá que o requisito de storage seja atendido.

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

## Use o Astra Trident com o Amazon FSX para NetApp ONTAP

"Amazon FSX para NetApp ONTAP"O , é um serviço AWS totalmente gerenciado que permite que os clientes iniciem e executem sistemas de arquivos equipados com o sistema operacional de storage ONTAP da NetApp. O Amazon FSX for NetApp ONTAP permite que você aproveite os recursos, o desempenho e os recursos administrativos do NetApp com os quais você já conhece, ao mesmo tempo em que aproveita a simplicidade, a agilidade, a segurança e a escalabilidade do armazenamento de dados na AWS. O FSX suporta muitos dos recursos do sistema de arquivos e APIs de administração do ONTAP.

Um sistema de arquivos é o principal recurso do Amazon FSX, análogo a um cluster do ONTAP no local. Em cada SVM, você pode criar um ou vários volumes, que são contentores de dados que armazenam os arquivos e pastas em seu sistema de arquivos. Com o Amazon FSX for NetApp ONTAP, o Data ONTAP será fornecido como um sistema de arquivos gerenciado na nuvem. O novo tipo de sistema de arquivos é chamado de **NetApp ONTAP**.

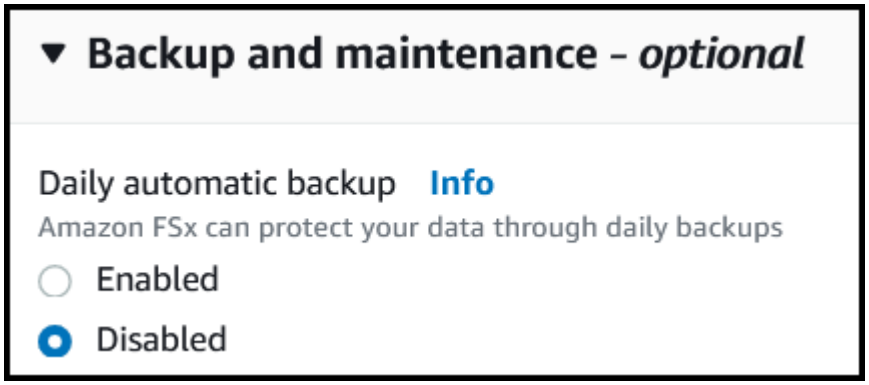
Usando o Astra Trident com o Amazon FSX for NetApp ONTAP, você pode garantir que os clusters do Kubernetes executados no Amazon Elastic Kubernetes Service (EKS) provisionem volumes persistentes de bloco e arquivo com o respaldo do do ONTAP.

**Criando seu sistema de arquivos do Amazon FSX for ONTAP**

Os volumes criados nos sistemas de arquivos do Amazon FSX que têm backups automáticos ativados não podem ser excluídos pelo Trident. Para excluir PVCs, você precisa excluir manualmente o PV e o volume FSX for ONTAP.

Para evitar este problema:

- Não use **Quick Create** para criar o sistema de arquivos FSX for ONTAP. O fluxo de trabalho de criação rápida permite backups automáticos e não fornece uma opção de exclusão.
- Ao usar **Standard Create**, desative o backup automático. A desativação de backups automáticos permite que o Trident exclua com êxito um volume sem intervenção manual adicional.




### Saiba mais sobre o Astra Trident

Se você é novo no Astra Trident, familiarize-se usando os links fornecidos abaixo:

- "FAQs"
- "Requisitos para uso do Astra Trident"
- "Implante o Astra Trident"

- "Práticas recomendadas para configurar o ONTAP, o Cloud Volumes ONTAP e o Amazon FSX for NetApp ONTAP"
- "Integre o Astra Trident"
- "Configuração de back-end SAN ONTAP"
- "Configuração de back-end do ONTAP nas"

Saiba mais sobre os recursos do "aqui"driver .

O Amazon FSX para NetApp ONTAP usa "FabricPool" para gerenciar camadas de armazenamento. Ele permite armazenar dados em um nível, com base no acesso frequente aos dados.

O Astra Trident espera ser executado como um `vsadmin` usuário SVM ou como um usuário com um nome diferente que tenha a mesma função. O Amazon FSX for NetApp ONTAP tem um `fsxadmin` usuário que é uma substituição limitada do usuário do cluster do ONTAP `admin`. Não é recomendável usar o `fsxadmin` usuário, com o Trident, pois `vsadmin` o usuário do SVM tem acesso a mais funcionalidades do Astra Trident.

#### Drivers

Você pode integrar o Astra Trident ao Amazon FSX for NetApp ONTAP usando os seguintes drivers:

- `ontap-san`: Cada PV provisionado é um LUN dentro de seu próprio volume do Amazon FSX for NetApp ONTAP.
- `ontap-san-economy`: Cada PV provisionado é um LUN com um número configurável de LUNs por volume do Amazon FSX for NetApp ONTAP.
- `ontap-nas`: Cada PV provisionado é um volume completo do Amazon FSX for NetApp ONTAP.
- `ontap-nas-economy`: Cada PV provisionado é uma qtree, com um número configurável de qtrees por volume do Amazon FSX for NetApp ONTAP.
- `ontap-nas-flexgroup`: Cada PV provisionado é um volume completo do Amazon FSX for NetApp ONTAP FlexGroup.

#### Autenticação

O Astra Trident oferece dois modos de autenticação:

- Baseado em certificado: O Astra Trident se comunicará com o SVM em seu sistema de arquivos FSX usando um certificado instalado no seu SVM.
- Baseado em credenciais: Você pode usar o `fsxadmin` usuário para o sistema de arquivos ou o `vsadmin` usuário configurado para o SVM.

> Recomendamos vivamente a utilização do `vsadmin` utilizador em vez do `fsxadmin` para configurar o back-end. O Astra Trident se comunicará com o sistema de arquivos FSX usando esse nome de usuário e senha.

Para saber mais sobre autenticação, consulte estes links:

- "ONTAP nas"
- "San ONTAP"

### Implante e configure o Astra Trident no EKS com o Amazon FSX for NetApp ONTAP

**O que você vai precisar**

- Um cluster do Amazon EKS existente ou um cluster do Kubernetes autogerenciado com `kubectl` o instalado.
- Um sistema de arquivos e uma máquina virtual de armazenamento (SVM) do Amazon FSX for NetApp ONTAP que pode ser acessado a partir dos nós de trabalho do seu cluster.
- Nós de trabalho preparados para "NFS e/ou iSCSI".

> Certifique-se de seguir as etapas de preparação de nós necessárias para o Amazon Linux e "Imagens de máquinas da Amazon" Ubuntu (AMIS), dependendo do seu tipo de AMI EKS.

Para outros requisitos do Astra Trident, "aqui"consulte .

**Passos**

1. Implante o Astra Trident usando um dos métodos de implantação../Trident-get-started/kupere-deploy.html.
2. Configure o Astra Trident da seguinte forma:
   a. Colete o nome DNS de LIF de gerenciamento do SVM. Por exemplo, usando a AWS CLI, localize a `DNSName` entrada em `Endpoints` → `Management` depois de executar o seguinte comando:

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

3. Criar e instalar certificados para autenticação. Se você estiver usando um `ontap-san` backend, "aqui"consulte . Se você estiver usando um `ontap-nas` backend, "aqui"consulte .

> Você pode fazer login no seu sistema de arquivos (por exemplo, para instalar certificados) usando SSH de qualquer lugar que possa chegar ao seu sistema de arquivos. Utilize o `fsxadmin` utilizador, a palavra-passe configurada quando criou o sistema de ficheiros e o nome DNS de gestão a partir `aws fsx describe-file-systems`do .

4. Crie um arquivo de back-end usando seus certificados e o nome DNS do seu LIF de gerenciamento, como mostrado na amostra abaixo:

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

Para obter informações sobre como criar backends, consulte estes links:

- "Configurar um back-end com drivers nas ONTAP"
- "Configure um back-end com drivers SAN ONTAP"

> Não especifique `dataLIF` para os `ontap-san` drivers e `ontap-san-economy` para permitir que o Astra Trident use multipath.

> O `limitAggregateUsage` parâmetro não funcionará com as `vsadmin` contas de utilizador e `fsxadmin` . A operação de configuração falhará se você especificar este parâmetro.

Após a implantação, execute as etapas para criar um "classe de storage, provisione um volume e monte o volume em um pod".

**Encontre mais informações**

- "Documentação do Amazon FSX para NetApp ONTAP"
- "Blog post no Amazon FSX for NetApp ONTAP"

# Crie backends com kubectl

Um back-end define a relação entre o Astra Trident e um sistema de storage. Ele diz ao Astra Trident como se comunicar com esse sistema de storage e como o Astra Trident deve provisionar volumes a partir dele. Após a instalação do Astra Trident, a próxima etapa é criar um back-end. A `TridentBackendConfig` Definição de recursos personalizada (CRD) permite criar e gerenciar backends Trident diretamente por meio da interface do Kubernetes. Para fazer isso, use `kubectl` ou a ferramenta CLI equivalente para sua distribuição do Kubernetes.

## `TridentBackendConfig`

`TridentBackendConfig`(`tbc tbconfig`, , `tbackendconfig`) É um CRD com namespaces e frontend que permite gerenciar backends Astra Trident usando `kubectl`o . Agora, os administradores de storage e Kubernetes podem criar e gerenciar back-ends diretamente pela CLI do Kubernetes sem exigir um utilitário de linha de comando dedicado (`tridentctl`).

Após a criação de `TridentBackendConfig` um objeto, acontece o seguinte:

- Um back-end é criado automaticamente pelo Astra Trident com base na configuração que você fornece. Isto é representado internamente como um `TridentBackend` (`tbe`, `tridentbackend` ) CR.
- O `TridentBackendConfig` é exclusivamente vinculado a um `TridentBackend` que foi criado pelo Astra Trident.

Cada `TridentBackendConfig` um mantém um mapeamento um-para-um com um `TridentBackend`. o primeiro é a interface fornecida ao usuário para projetar e configurar backends; o último é como o Trident representa o objeto backend real.

> `TridentBackend` Os CRS são criados automaticamente pelo Astra Trident. Você **não deve** modificá-los. Se você quiser fazer atualizações para backends, faça isso modificando o `TridentBackendConfig` objeto.

Veja o exemplo a seguir para o formato do `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
```

Você também pode dar uma olhada nos exemplos "instalador do Trident" no diretório para configurações de exemplo para a plataforma/serviço de armazenamento desejado.

O `spec` utiliza parâmetros de configuração específicos do back-end. Neste exemplo, o backend usa o `ontap-san` driver de armazenamento e usa os parâmetros de configuração que são tabulados aqui. Para obter a lista de opções de configuração do driver de armazenamento desejado, consulte "informações de configuração de back-end para seu driver de armazenamento".

A `spec` seção também inclui `credentials` campos e `deletionPolicy`, que são recentemente introduzidos no `TridentBackendConfig` CR:

- `credentials`: Este parâmetro é um campo obrigatório e contém as credenciais usadas para autenticar com o sistema/serviço de armazenamento. Isso é definido como um segredo do Kubernetes criado pelo usuário. As credenciais não podem ser passadas em texto simples e resultarão em um erro.
- `deletionPolicy`: Este campo define o que deve acontecer quando o `TridentBackendConfig` é excluído. Pode tomar um dos dois valores possíveis:
  - `delete`: Isso resulta na exclusão do `TridentBackendConfig` CR e do back-end associado. Este é o valor padrão.
  - `retain`: Quando um `TridentBackendConfig` CR é excluído, a definição de back-end ainda estará presente e poderá ser gerenciada com `tridentctl`o . Definir a política de exclusão para `retain` permitir que os usuários façam o downgrade para uma versão anterior (anterior a 21,04) e mantenham os backends criados. O valor para este campo pode ser atualizado após a criação de um `TridentBackendConfig`.

> ⓘ O nome de um back-end é definido usando `spec.backendName`. Se não for especificado, o nome do backend é definido como o nome do `TridentBackendConfig` objeto (metadata.name). Recomenda-se definir explicitamente nomes de back-end usando `spec.backendName`o .

Backends que foram criados com `tridentctl` não têm um objeto associado `TridentBackendConfig`. Você pode optar por gerenciar esses backends `kubectl` criando um `TridentBackendConfig` CR. Deve-se ter cuidado para especificar parâmetros de configuração idênticos (como `spec.backendName` , , `spec.storagePrefix`, `spec.storageDriverName` e assim por diante). O Astra Trident vinculará automaticamente o recém-criado `TridentBackendConfig` ao back-end pré-existente.

## Visão geral dos passos

Para criar um novo back-end usando `kubectl`o , você deve fazer o seguinte:

1. Criar um "Segredo do Kubernetes". o segredo contém as credenciais que o Astra Trident precisa para se comunicar com o cluster/serviço de storage.
2. Crie `TridentBackendConfig` um objeto. Isso contém detalhes sobre o cluster/serviço de armazenamento e faz referência ao segredo criado na etapa anterior.

Depois de criar um backend, você pode observar seu status usando `kubectl get tbc <tbc-name> -n <trident-namespace>` e coletar detalhes adicionais.

## Etapa 1: Crie um segredo do Kubernetes

Crie um segredo que contenha as credenciais de acesso para o back-end. Isso é exclusivo para cada serviço/plataforma de storage. Aqui está um exemplo:

```
$ 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(>
```

Esta tabela resume os campos que devem ser incluídos no segredo para cada plataforma de armazenamento:

| Descrição dos campos secretos da plataforma de armazenamento | Segredo | Descrição dos campos |
|---|---|---|
| Azure NetApp Files | ID do cliente | A ID do cliente a partir de um registo de aplicação |
| Cloud Volumes Service para GCP | private_key_id | ID da chave privada. Parte da chave da API para a conta de serviço do GCP com a função de administrador do CVS |

| Descrição dos campos secretos da plataforma de armazenamento | Segredo | Descrição dos campos |
|---|---|---|
| Cloud Volumes Service para GCP | chave_privada | Chave privada. Parte da chave da API para a conta de serviço do GCP com a função de administrador do CVS |
| Elemento (NetApp HCI/SolidFire) | Endpoint | MVIP para o cluster SolidFire com credenciais de locatário |
| ONTAP | nome de utilizador | Nome de usuário para se conetar ao cluster/SVM. Usado para autenticação baseada em credenciais |
| ONTAP | palavra-passe | Senha para se conectar ao cluster/SVM. Usado para autenticação baseada em credenciais |
| ONTAP | ClientPrivateKey | Valor codificado em base64 da chave privada do cliente. Usado para autenticação baseada em certificado |
| ONTAP | ChapUsername | Nome de utilizador de entrada. Necessário se useCHAP-true. Para `ontap-san` e. `ontap-san-economy` |
| ONTAP | IniciadorSecreto | Segredo do iniciador CHAP. Necessário se useCHAP-true. Para `ontap-san` e. `ontap-san-economy` |
| ONTAP | ChapTargetUsername | Nome de utilizador alvo. Necessário se useCHAP-true. Para `ontap-san` e. `ontap-san-economy` |
| ONTAP | ChapTargetInitiatorSecret | Segredo do iniciador de destino CHAP. Necessário se useCHAP-true. Para `ontap-san` e. `ontap-san-economy` |

O segredo criado nesta etapa será referenciado `spec.credentials` no campo do `TridentBackendConfig` objeto que é criado na próxima etapa.

## Passo 2: Crie o `TridentBackendConfig` CR

Agora você está pronto para criar seu `TridentBackendConfig` CR. Neste exemplo, um back-end que usa `ontap-san` o driver é criado usando o `TridentBackendConfig` objeto mostrado abaixo:

```
$ 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
```

## Etapa 3: Verifique o status do `TridentBackendConfig` CR

Agora que criou o `TridentBackendConfig` CR, pode verificar o estado. Veja o exemplo a seguir:

```
$ 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
```

Um backend foi criado com sucesso e vinculado ao `TridentBackendConfig` CR.

A fase pode ter um dos seguintes valores:

- `Bound`: O `TridentBackendConfig` CR está associado a um back-end, e esse backend contém `configRef` definido como `TridentBackendConfig` UID do CR.
- `Unbound`: Representado `""` usando . O `TridentBackendConfig` objeto não está vinculado a um backend. Todos os CRS recém-criados `TridentBackendConfig` estão nesta fase por padrão. Após as alterações de fase, ela não pode voltar a Unbound.
- `Deleting`: Os `TridentBackendConfig` CR `deletionPolicy` foram definidos para eliminar. Quando o `TridentBackendConfig` CR é excluído, ele passa para o estado de exclusão.
  - Se não houver declarações de volume persistentes (PVCs) no back-end, a exclusão do resultará na

exclusão do `TridentBackendConfig` Astra Trident do back-end e do `TridentBackendConfig` CR.

    - Se um ou mais PVCs estiverem presentes no back-end, ele vai para um estado de exclusão. Posteriormente, o `TridentBackendConfig` CR entra também na fase de eliminação. O back-end e `TridentBackendConfig` são excluídos somente depois que todos os PVCs são excluídos.

- `Lost`: O back-end associado ao `TridentBackendConfig` CR foi acidentalmente ou deliberadamente excluído e o `TridentBackendConfig` CR ainda tem uma referência ao back-end excluído. O `TridentBackendConfig` CR ainda pode ser eliminado independentemente do `deletionPolicy` valor.
- `Unknown`: O Astra Trident não consegue determinar o estado ou a existência do back-end associado ao `TridentBackendConfig` CR. Por exemplo, se o servidor de API não estiver respondendo ou se o `tridentbackends.trident.netapp.io` CRD estiver ausente. Isso pode exigir a intervenção do usuário.

Nesta fase, um backend é criado com sucesso! Existem várias operações que podem ser tratadas adicionalmente, "atualizações de back-end e exclusões de back-end"como o .

## (Opcional) passo 4: Obtenha mais detalhes

Você pode executar o seguinte comando para obter mais informações sobre seu back-end:

```
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
```

Além disso, você também pode obter um despejo YAML/JSON do `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` Contém o `backendName` e o `backendUUID` do back-end criado em resposta ao `TridentBackendConfig` CR. O `lastOperationStatus` campo representa o status da última operação `TridentBackendConfig` do CR, que pode ser acionada pelo usuário (por exemplo, o usuário mudou algo no `spec`) ou acionada pelo Astra Trident (por exemplo, durante reinicializações do Astra Trident). Pode ser sucesso ou falhou. `phase` Representa o status da relação entre o `TridentBackendConfig` CR e o back-end. No exemplo acima, `phase` tem o valor vinculado, o que significa que o `TridentBackendConfig` CR está associado ao back-end.

Você pode executar o `kubectl -n trident describe tbc <tbc-cr-name>` comando para obter detalhes dos logs de eventos.

Não é possível atualizar ou excluir um back-end que contenha um objeto `tridentctl` associado `TridentBackendConfig` usando o . Compreender as etapas envolvidas na troca entre `tridentctl` e `TridentBackendConfig`, "veja aqui".

# Execute o gerenciamento de back-end com o kubectl

Saiba mais sobre como executar operações de gerenciamento de back-end usando `kubectl`o .

## Excluir um back-end

Ao excluir um `TridentBackendConfig`, você instrui o Astra Trident a excluir/reter backends (com base `deletionPolicy` no ). Para excluir um back-end, certifique-se de que `deletionPolicy` está definido para excluir. Para eliminar apenas o `TridentBackendConfig`, certifique-se de que `deletionPolicy` está definido como reter. Isso garantirá que o backend ainda esteja presente e possa ser gerenciado usando `tridentctl`o .

Execute o seguinte comando:

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

O Astra Trident não exclui os segredos do Kubernetes que estavam em uso `TridentBackendConfig` pelo . O usuário do Kubernetes é responsável pela limpeza de segredos. Cuidado deve ser tomado ao excluir segredos. Você deve excluir segredos somente se eles não estiverem em uso pelos backends.

## Veja os backends existentes

Execute o seguinte comando:

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

Você também pode executar `tridentctl get backend -n trident` ou `tridentctl get backend -o yaml -n trident` obter uma lista de todos os backends que existem. Esta lista também incluirá backends que foram criados com `tridentctl`.

## Atualize um back-end

Pode haver várias razões para atualizar um backend:

- As credenciais para o sistema de storage foram alteradas. Para atualizar as credenciais, o segredo do Kubernetes que é usado no `TridentBackendConfig` objeto deve ser atualizado. O Astra Trident atualizará automaticamente o back-end com as credenciais mais recentes fornecidas. Execute o seguinte comando para atualizar o segredo do Kubernetes:

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

- Os parâmetros (como o nome do SVM do ONTAP sendo usado) precisam ser atualizados. Nesse caso, `TridentBackendConfig` os objetos podem ser atualizados diretamente pelo Kubernetes.

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

Alternativamente, faça alterações no CR existente `TridentBackendConfig` executando o seguinte comando:

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

Se uma atualização de back-end falhar, o back-end continuará em sua última configuração conhecida. Pode visualizar os registos para determinar a causa executando `kubectl get tbc <tbc-name> -o yaml -n trident` ou `kubectl describe tbc <tbc-name> -n trident`.

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode executar novamente o comando update.

# Execute o gerenciamento de back-end com o tridentctl

Saiba mais sobre como executar operações de gerenciamento de back-end usando `tridentctl`o .

## Crie um backend

Depois de criar um "arquivo de configuração de back-end", execute o seguinte comando:

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

Se a criação do backend falhar, algo estava errado com a configuração do backend. Você pode exibir os logs para determinar a causa executando o seguinte comando:

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

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode simplesmente executar o `create` comando novamente.

## Excluir um back-end

Para excluir um back-end do Astra Trident, faça o seguinte:

1. Recuperar o nome do backend:

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

2. Excluir o backend:

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

Se o Astra Trident provisionou volumes e snapshots desse back-end que ainda existem, a exclusão do back-end impede que novos volumes sejam provisionados por ele. O back-end continuará a existir em um estado de exclusão e o Trident continuará a gerenciar esses volumes e snapshots até que sejam excluídos.

## Veja os backends existentes

Para visualizar os backends que o Trident conhece, faça o seguinte:

- Para obter um resumo, execute o seguinte comando:

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

- Para obter todos os detalhes, execute o seguinte comando:

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

## Atualize um back-end

Depois de criar um novo arquivo de configuração de back-end, execute o seguinte comando:

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

Se a atualização do backend falhar, algo estava errado com a configuração do backend ou você tentou uma atualização inválida. Você pode exibir os logs para determinar a causa executando o seguinte comando:

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

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode simplesmente executar o `update` comando novamente.

## Identificar as classes de armazenamento que usam um back-end

Este é um exemplo do tipo de perguntas que você pode responder com o JSON que `tridentctl` produz para objetos de back-end. Isso usa o `jq` utilitário, que você precisa instalar.

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

Isso também se aplica a backends que foram criados usando `TridentBackendConfig`o .

# Alternar entre opções de gerenciamento de back-end

Saiba mais sobre as diferentes maneiras de gerenciar back-ends no Astra Trident. Com a introdução `TridentBackendConfig`do , os administradores agora têm duas maneiras exclusivas de gerenciar backends. Isso coloca as seguintes perguntas:

- Os backends podem ser criados usando `tridentctl` ser gerenciados com `TridentBackendConfig`?
- Os backends podem ser criados usando `TridentBackendConfig` ser gerenciados `tridentctl` usando ?

## Gerenciar `tridentctl` backends usando `TridentBackendConfig`

Esta seção aborda as etapas necessárias para gerenciar backends que foram criados usando `tridentctl` diretamente a interface do Kubernetes criando `TridentBackendConfig` objetos.

Isso se aplicará aos seguintes cenários:

- Backends pré-existentes, que não têm um `TridentBackendConfig` porque foram criados com `tridentctl`.
- Novos backends que foram criados com `tridentctl`, enquanto outros `TridentBackendConfig` objetos existem.

Em ambos os cenários, os back-ends continuarão presentes, com o Astra Trident agendando volumes e operando neles. Os administradores têm uma das duas opções aqui:

- Continue `tridentctl` usando para gerenciar backends que foram criados usando-o.
- Vincular backends criados usando `tridentctl` a um novo `TridentBackendConfig` objeto. Fazer isso significaria que os backends serão gerenciados usando `kubectl` e não `tridentctl`.

Para gerenciar um back-end pré-existente usando `kubectl`o , você precisará criar um `TridentBackendConfig` que se vincule ao back-end existente. Aqui está uma visão geral de como isso funciona:

1. Crie um segredo do Kubernetes. O segredo contém as credenciais que o Astra Trident precisa para se comunicar com o cluster/serviço de storage.
2. Crie `TridentBackendConfig` um objeto. Isso contém detalhes sobre o cluster/serviço de armazenamento e faz referência ao segredo criado na etapa anterior. Deve-se ter cuidado para especificar parâmetros de configuração idênticos (como `spec.backendName` , , `spec.storagePrefix`, `spec.storageDriverName` e assim por diante). `spec.backendName` deve ser definido como o nome do backend existente.

### Passo 0: Identifique o backend

Para criar um `TridentBackendConfig` que se vincula a um backend existente, você precisará obter a configuração do backend. Neste exemplo, vamos supor que um backend foi criado usando a seguinte definição 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"
            }
```

```
        }
    ]
}
```

**Etapa 1: Crie um segredo do Kubernetes**

Crie um segredo que contenha as credenciais para o back-end, como mostrado neste exemplo:

```
$ 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
```

**Passo 2: Crie um `TridentBackendConfig` CR**

O próximo passo é criar um `TridentBackendConfig` CR que se vinculará automaticamente ao pré-existente `ontap-nas-backend` (como neste exemplo). Certifique-se de que os seguintes requisitos são cumpridos:

- O mesmo nome de back-end é definido no `spec.backendName`.
- Os parâmetros de configuração são idênticos ao back-end original.
- Os pools de armazenamento virtual (se presentes) devem manter a mesma ordem que no back-end original.
- As credenciais são fornecidas por meio de um segredo do Kubernetes e não em texto simples.

Neste caso, o `TridentBackendConfig` será parecido com este:

```
$ 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
```

**Etapa 3: Verifique o status do `TridentBackendConfig` CR**

Após a criação do `TridentBackendConfig` , sua fase deve ser `Bound`. Ele também deve refletir o mesmo nome de back-end e UUID que o do back-end existente.

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

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

O backend agora será completamente gerenciado usando o `tbc-ontap-nas-backend TridentBackendConfig` objeto.

## Gerenciar `TridentBackendConfig` backends usando `tridentctl`

```
`tridentctl` pode ser usado para listar backends que foram criados usando
`TridentBackendConfig`. Além disso, os administradores também podem optar
por gerenciar completamente esses backends `tridentctl` excluindo
`TridentBackendConfig` e certificando-se de `spec.deletionPolicy` que está
definido como `retain`.
```

### Passo 0: Identifique o backend

Por exemplo, vamos supor que o seguinte backend foi criado usando `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 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

A partir da saída, vê-se que `TridentBackendConfig` foi criado com sucesso e está vinculado a um backend [observe o UUID do backend].

**Passo 1: Confirmar** `deletionPolicy` **está definido como** `retain`

Vamos dar uma olhada no valor `deletionPolicy` de . Isso precisa ser definido como `retain`. Isso garantirá que, quando um `TridentBackendConfig` CR for excluído, a definição de back-end ainda estará presente e poderá ser gerenciada com `tridentctl`o .

```
$ 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
```

Não avance para o passo seguinte, a menos `deletionPolicy` que esteja definido para `retain`.

**Etapa 2: Exclua o `TridentBackendConfig` CR**

O passo final é eliminar o `TridentBackendConfig` CR. Depois de confirmar que o `deletionPolicy` está definido como `retain`, pode avançar com a eliminação:

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

Após a exclusão `TridentBackendConfig` do objeto, o Astra Trident simplesmente o remove sem realmente excluir o próprio back-end.

# Gerenciar classes de armazenamento

Encontre informações sobre como criar uma classe de armazenamento, excluir uma classe de armazenamento e exibir classes de armazenamento existentes.

## Crie uma classe de armazenamento

Consulte "aqui" para obter mais informações sobre quais são as classes de armazenamento e como as configura.

## Crie uma classe de armazenamento

Depois de ter um arquivo de classe de armazenamento, execute o seguinte comando:

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

`<storage-class-file>` deve ser substituído pelo nome do arquivo da classe de armazenamento.

## Excluir uma classe de armazenamento

Para excluir uma classe de armazenamento do Kubernetes, execute o seguinte comando:

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

`<storage-class>` deve ser substituído pela sua classe de armazenamento.

Todos os volumes persistentes criados com essa classe de storage permanecerão intocados, e o Astra Trident continuará gerenciá-los.

> O Astra Trident impõe um espaço em branco `fsType` para os volumes que cria. Para backends iSCSI, recomenda-se aplicar `parameters.fsType` no StorageClass. Você deve excluir o esixting StorageClasses e recriá-los com `parameters.fsType` o especificado.

## Exibir as classes de armazenamento existentes

- Para visualizar as classes de armazenamento do Kubernetes existentes, execute o seguinte comando:

```
kubectl get storageclass
```

- Para ver os detalhes da classe de storage do Kubernetes, execute o seguinte comando:

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

- Para exibir as classes de storage sincronizadas do Astra Trident, execute o seguinte comando:

```
tridentctl get storageclass
```

- Para visualizar os detalhes da classe de storage sincronizado do Astra Trident, execute o seguinte comando:

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

## Defina uma classe de armazenamento padrão

O Kubernetes 1,6 adicionou a capacidade de definir uma classe de storage padrão. Esta é a classe de armazenamento que será usada para provisionar um volume persistente se um usuário não especificar um em uma reivindicação de volume persistente (PVC).

- Defina uma classe de armazenamento padrão definindo a anotação `storageclass.kubernetes.io/is-default-class` como verdadeira na definição da classe de armazenamento. De acordo com a especificação, qualquer outro valor ou ausência da anotação é interpretado como falso.

- Você pode configurar uma classe de armazenamento existente para ser a classe de armazenamento padrão usando o seguinte comando:

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

- Da mesma forma, você pode remover a anotação de classe de armazenamento padrão usando o seguinte comando:

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

Há também exemplos no pacote de instalação do Trident que incluem esta anotação.

> Você deve ter apenas uma classe de armazenamento padrão em seu cluster a qualquer momento. O Kubernetes não impede tecnicamente que você tenha mais de um, mas se comportará como se não houvesse nenhuma classe de storage padrão.

### Identificar o back-end de uma classe de storage

Este é um exemplo do tipo de perguntas que você pode responder com o JSON que `tridentctl` produz para objetos backend Astra Trident. Isso usa o `jq` utilitário, que você pode precisar instalar primeiro.

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

# Executar operações de volume

Saiba mais sobre os recursos fornecidos pelo Astra Trident para gerenciar seus volumes.

- "Use a topologia CSI"
- "Trabalhar com instantâneos"
- "Expanda volumes"
- "Importar volumes"

## Use a topologia CSI

O Astra Trident pode criar e anexar volumes de forma seletiva a nós presentes em um cluster Kubernetes usando o "Recurso de topologia CSI". Usando o recurso de topologia de CSI, o acesso a volumes pode ser limitado a um subconjunto de nós, com base em regiões e zonas de disponibilidade. Hoje em dia, os provedores de nuvem permitem que os administradores do Kubernetes gerem nós baseados em zonas. Os nós podem ser localizados em diferentes zonas de disponibilidade dentro de uma região ou em várias regiões. Para facilitar o provisionamento de volumes para workloads em uma arquitetura de várias zonas, o Astra Trident usa topologia de CSI.

Saiba mais sobre o recurso de topologia de CSI "aqui" .

O Kubernetes oferece dois modos exclusivos de vinculação de volume:

- `VolumeBindingMode`Com o definido como `Immediate`, o Astra Trident cria o volume sem qualquer reconhecimento de topologia. A vinculação de volume e o provisionamento dinâmico são tratados quando o PVC é criado. Esse é o padrão `VolumeBindingMode` e é adequado para clusters que não impõem restrições de topologia. Os volumes persistentes são criados sem depender dos requisitos de agendamento do pod solicitante.
- Com `VolumeBindingMode` definido como `WaitForFirstConsumer`, a criação e a vinculação de um volume persistente para um PVC é adiada até que um pod que usa o PVC seja programado e criado. Dessa forma, os volumes são criados para atender às restrições de agendamento impostas pelos requisitos de topologia.

O `WaitForFirstConsumer` modo de encadernação não requer rótulos de topologia. Isso pode ser usado independentemente do recurso de topologia de CSI.

**O que você vai precisar**

Para fazer uso da topologia de CSI, você precisa do seguinte:

- Um cluster do Kubernetes executando o 1,17 ou posterior.

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

- Os nós no cluster devem ter rótulos que introduzam reconhecimento da topologia (`topology.kubernetes.io/region`e `topology.kubernetes.io/zone`). Esses rótulos **devem estar presentes nos nós no cluster** antes que o Astra Trident seja instalado para que o Astra Trident esteja ciente da topologia.

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

**Etapa 1: Crie um back-end com reconhecimento de topologia**

Os back-ends de storage do Astra Trident podem ser desenvolvidos para provisionar volumes de forma seletiva, com base nas zonas de disponibilidade. Cada back-end pode transportar um bloco opcional `supportedTopologies` que representa uma lista de zonas e regiões que devem ser suportadas. Para o StorageClasses que fazem uso de tal back-end, um volume só seria criado se solicitado por um aplicativo agendado em uma região/zona suportada.

Veja como é um exemplo de definição de backend:

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

`supportedTopologies` é usado para fornecer uma lista de regiões e zonas por backend. Essas regiões e zonas representam a lista de valores permitidos que podem ser fornecidos em um StorageClass. Para os StorageClasses que contêm um subconjunto das regiões e zonas fornecidas em um back-end, o Astra Trident criará um volume no back-end.

Você também pode definir `supportedTopologies` por pool de armazenamento. Veja o exemplo a seguir:

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

Neste exemplo, as `region` etiquetas e `zone` representam a localização do conjunto de armazenamento. `topology.kubernetes.io/region` `topology.kubernetes.io/zone` e dite de onde os pools de storage podem ser consumidos.

#### Etapa 2: Defina StorageClasses que estejam cientes da topologia

Com base nas etiquetas de topologia fornecidas aos nós no cluster, o StorageClasses pode ser definido para conter informações de topologia. Isso determinará os pools de storage que atuam como candidatos a solicitações de PVC feitas e o subconjunto de nós que podem fazer uso dos volumes provisionados pelo Trident.

Veja o exemplo a seguir:

```
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"
```

Na definição StorageClass fornecida acima, `volumeBindingMode` está definida como `WaitForFirstConsumer`. Os PVCs solicitados com este StorageClass não serão utilizados até que sejam referenciados em um pod. E, `allowedTopologies` fornece as zonas e a região a serem usadas. O `netapp-san-us-east1` StorageClass criará PVCs no `san-backend-us-east1` back-end definido acima.

**Passo 3: Criar e usar um PVC**

Com o StorageClass criado e mapeado para um back-end, agora você pode criar PVCs.

Veja o exemplo `spec` abaixo:

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

Criar um PVC usando este manifesto resultaria no seguinte:

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

Para o Trident criar um volume e vinculá-lo ao PVC, use o PVC em um pod. Veja o exemplo a seguir:

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

Este podSpec instrui o Kubernetes a agendar o pod em nós presentes na `us-east1` região e escolher entre qualquer nó presente nas `us-east1-a` zonas ou `us-east1-b`.

Veja a seguinte saída:

```
$ 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
```

**Atualize os backends para incluir** `supportedTopologies`

Os backends pré-existentes podem ser atualizados para incluir uma lista `supportedTopologies` de uso `tridentctl backend update`do . Isso não afetará os volumes que já foram provisionados e só será usado para PVCs subsequentes.

**Encontre mais informações**

- "Gerenciar recursos para contêineres"
- "NodeSeletor"
- "Afinidade e anti-afinidade"
- "Taints e Tolerations"

## Trabalhar com instantâneos

A partir do lançamento de 20,01 do Astra Trident, você pode criar snapshots de PVS na camada Kubernetes. Use esses snapshots para manter cópias pontuais de volumes criados pelo Astra Trident e agendar a criação de volumes adicionais (clones). O instantâneo de volume é suportado pelos `ontap-nas` drivers , `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs` e `azure-netapp-files` .

Esse recurso está disponível no Kubernetes 1,17 (beta) e é GA no 1,20. Para entender as mudanças envolvidas na mudança de beta para GA, "o blog de lançamento" consulte . Com a graduação para GA, a `v1` versão da API é introduzida e é compatível com `v1beta1` snapshots.

**O que você vai precisar**

- A criação de instantâneos de volume requer a criação de um controlador de instantâneos externo, bem como de algumas CRDs (Custom Resource Definitions). Essa é a responsabilidade do orquestrador do Kubernetes que está sendo usado (por exemplo: Kubeadm, GKE, OpenShift).

Você pode criar um controlador de snapshot externo e CRDs de snapshot da seguinte forma:

1. Criar CRDs de instantâneos de volume:

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

2. Crie o snapshot-controller no namespace desejado. Edite os manifestos YAML abaixo para modificar o namespace.

Não crie um controlador instantâneo se configurar instantâneos de volume sob demanda em um ambiente GKE. O GKE utiliza um controlador instantâneo oculto incorporado.

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

O CSI Snapshotter fornece um "validar webhook" para ajudar os usuários a validar snapshots existentes do v1beta1 e confirmar que são objetos de recurso válidos. O webhook de validação rotula automaticamente objetos snapshot inválidos e impede a criação de futuros objetos inválidos. O webhook de validação é implantado pelo Kubernetes orchestrator. Consulte as instruções para implantar o webhook de validação manualmente "aqui". Encontre exemplos de manifestos de instantâneos inválidos "aqui" .

O exemplo detalhado abaixo explica as construções necessárias para trabalhar com snapshots e mostra como os snapshots podem ser criados e usados.

**Passo 1: Configure a.** `VolumeSnapshotClass`

Antes de criar um instantâneo de volume, configure um xref:./trident-use/../Trident-reference/objects.html[`VolumeSnapshotClass`.

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

```
`driver`O ponto é o condutor CSI do Astra Trident. `deletionPolicy` pode
ser `Delete` ou `Retain`. Quando definido como `Retain`, o instantâneo
físico subjacente no cluster de armazenamento é retido mesmo quando o
`VolumeSnapshot` objeto é excluído.
```

**Passo 2: Crie um instantâneo de um PVC existente**

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

O instantâneo está sendo criado para um PVC chamado `pvc1`, e o nome do instantâneo é definido como `pvc1-snap`.

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

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

Isso criou um `VolumeSnapshot` objeto. Um VolumeSnapshot é análogo a um PVC e está associado a um `VolumeSnapshotContent` objeto que representa o snapshot real.

É possível identificar o `VolumeSnapshotContent` objeto para o `pvc1-snap` VolumeSnapshot descrevendo-o.

```
$ 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
.
.
```

O `Snapshot Content Name` identifica o objeto VolumeSnapshotContent que serve este instantâneo. O `Ready To Use` parâmetro indica que o instantâneo pode ser usado para criar um novo PVC.

**Etapa 3: Criar PVCs a partir do VolumeSnapshots**

Veja o exemplo a seguir para criar um PVC usando um snapshot:

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

`dataSource` Mostra que o PVC deve ser criado usando um VolumeSnapshot nomeado `pvc1-snap` como a fonte dos dados. Isso instrui o Astra Trident a criar um PVC a partir do snapshot. Depois que o PVC é criado, ele pode ser anexado a um pod e usado como qualquer outro PVC.

> Ao excluir um volume persistente com snapshots associados, o volume Trident correspondente é atualizado para um "estado de exclusão". Para que o volume do Astra Trident seja excluído, os snapshots do volume devem ser removidos.

#### Encontre mais informações

- "Instantâneos de volume"
- xref:./trident-use/../Trident-reference/objects.html[`VolumeSnapshotClass`

## Expanda volumes

O Astra Trident oferece aos usuários do Kubernetes a capacidade de expandir seus volumes depois que eles são criados. Encontre informações sobre as configurações necessárias para expandir volumes iSCSI e NFS.

#### Expanda um volume iSCSI

É possível expandir um iSCSI Persistent volume (PV) usando o provisionador de CSI.

> A expansão de volume iSCSI é suportada pelos `ontap-san` `ontap-san-economy` drivers , , `solidfire-san` e requer o Kubernetes 1,16 e posterior.

##### Visão geral

A expansão de um iSCSI PV inclui os seguintes passos:

- Editando a definição StorageClass para definir o `allowVolumeExpansion` campo como `true`.
- Editar a definição de PVC e atualizar o `spec.resources.requests.storage` para refletir o tamanho

recém-desejado, que deve ser maior que o tamanho original.

- A fixação do PV deve ser fixada a um pod para que ele seja redimensionado. Existem dois cenários ao redimensionar um iSCSI PV:
  - Se o PV estiver conetado a um pod, o Astra Trident expande o volume no back-end de armazenamento, refaz o dispositivo e redimensiona o sistema de arquivos.
  - Ao tentar redimensionar um PV não anexado, o Astra Trident expande o volume no back-end de armazenamento. Depois que o PVC é ligado a um pod, o Trident refaz o dispositivo e redimensiona o sistema de arquivos. Em seguida, o Kubernetes atualiza o tamanho do PVC após a operação de expansão ter sido concluída com sucesso.

O exemplo abaixo mostra como funcionam os PVS iSCSI em expansão.

#### Etapa 1: Configure o StorageClass para dar suporte à expansão de volume

```
$ 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
```

Para um StorageClass já existente, edite-o para incluir o `allowVolumeExpansion` parâmetro.

#### Etapa 2: Crie um PVC com o StorageClass que você criou

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

O Astra Trident cria um volume persistente (PV) e o associa a essa reivindicação de volume persistente (PVC).

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

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

#### Passo 3: Defina um pod que prende o PVC

Neste exemplo, é criado um pod que usa o `san-pvc`.

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

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

#### Passo 4: Expanda o PV

Para redimensionar o PV que foi criado de 1Gi a 2Gi, edite a definição de PVC e atualize o `spec.resources.requests.storage` para 2Gi.

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

**Etapa 5: Validar a expansão**

É possível validar a expansão trabalhada corretamente verificando o tamanho do PVC, PV e volume Astra Trident:

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

### Expandir um volume NFS

O Astra Trident dá suporte à expansão de volume para PVS NFS provisionados em `ontap-nas` `ontap-nas-economy` , , , `ontap-nas-flexgroup` `gcp-cvs` e `azure-netapp-files` backends.

#### Etapa 1: Configure o StorageClass para dar suporte à expansão de volume

Para redimensionar um PV NFS, o administrador primeiro precisa configurar a classe de armazenamento para permitir a expansão de volume definindo o `allowVolumeExpansion` campo para `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 você já criou uma classe de armazenamento sem essa opção, você pode simplesmente editar a classe de armazenamento existente usando `kubectl edit storageclass` para permitir a expansão de volume.

**Etapa 2: Crie um PVC com o StorageClass que você criou**

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

O Astra Trident deve criar um PV NFS de 20MiB para este 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
```

**Passo 3: Expanda o PV**

Para redimensionar o 20MiB PV recém-criado para 1GiB, edite o PVC e defina `spec.resources.requests.storage` como 1GB:

```
$ 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
...
```

**Etapa 4: Validar a expansão**

Você pode validar o redimensionamento trabalhado corretamente verificando o tamanho do PVC, PV e o volume Astra Trident:

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

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

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

## Importar volumes

Você pode importar volumes de armazenamento existentes como um PV do Kubernetes usando `tridentctl import`o .

#### Drivers que suportam importação de volume

Esta tabela mostra os drivers que suportam a importação de volumes e a versão em que foram introduzidos.

| Condutor | Solte |
|---|---|
| `ontap-nas` | 19,04 |
| `ontap-nas-flexgroup` | 19,04 |
| `solidfire-san` | 19,04 |
| `azure-netapp-files` | 19,04 |

| Condutor | Solte |
| --- | --- |
| `gcp-cvs` | 19,04 |
| `ontap-san` | 19,04 |

**Por que devo importar volumes?**

Existem vários casos de uso para importar um volume para o Trident:

- Conteinerizar um aplicativo e reutilizar seu conjunto de dados existente
- Usando um clone de um conjunto de dados para uma aplicação efêmera
- Reconstruindo um cluster do Kubernetes com falha
- Migração de dados de aplicativos durante a recuperação de desastres

**Como funciona a importação?**

O arquivo PVC (Persistent volume Claim) é usado pelo processo de importação de volume para criar o PVC. No mínimo, o arquivo PVC deve incluir os campos nome, namespace, accessModes e storageClassName como mostrado no exemplo a seguir.

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

O `tridentctl` cliente é usado para importar um volume de armazenamento existente. O Trident importa o volume persistindo metadados de volume e criando o PVC e o PV.

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

Para importar um volume de storage, especifique o nome do back-end do Astra Trident que contém o volume, bem como o nome que identifica exclusivamente o volume no storage (por exemplo: ONTAP FlexVol, Element volume, caminho de volume CVS). O volume de storage deve permitir acesso de leitura/gravação e ser acessível pelo back-end especificado do Astra Trident. O `-f` argumento string é necessário e especifica o caminho para o arquivo PVC YAML ou JSON.

Quando o Astra Trident recebe a solicitação de volume de importação, o tamanho do volume existente é determinado e definido no PVC. Depois que o volume é importado pelo driver de armazenamento, o PV é criado com uma ClaimRef para o PVC. A política de recuperação é inicialmente definida como `retain` no PV.

Depois que o Kubernetes vincula com êxito o PVC e o PV, a política de recuperação é atualizada para corresponder à política de recuperação da Classe de armazenamento. Se a política de recuperação da Classe de armazenamento for `delete`, o volume de armazenamento será excluído quando o PV for excluído.

Quando um volume é importado com o `--no-manage` argumento, o Trident não executa nenhuma operação adicional no PVC ou PV para o ciclo de vida dos objetos. Como o Trident ignora eventos PV e PVC para `--no-manage` objetos, o volume de armazenamento não é excluído quando o PV é excluído. Outras operações, como clone de volume e redimensionamento de volume, também são ignoradas. Essa opção é útil se você quiser usar o Kubernetes para workloads em contêineres, mas de outra forma quiser gerenciar o ciclo de vida do volume de storage fora do Kubernetes.

Uma anotação é adicionada ao PVC e ao PV que serve para um duplo propósito de indicar que o volume foi importado e se o PVC e o PV são gerenciados. Esta anotação não deve ser modificada ou removida.

O Trident 19,07 e posterior lidam com a fixação de PVS e monta o volume como parte da importação. Para importações usando versões anteriores do Astra Trident, não haverá nenhuma operação no caminho de dados e a importação de volume não verificará se o volume pode ser montado. Se um erro for cometido com a importação de volume (por exemplo, o StorageClass está incorreto), você poderá recuperar alterando a política de recuperação no PV para `retain`, excluindo o PVC e o PV e tentando novamente o comando de importação de volume.

### `ontap-nas` e `ontap-nas-flexgroup` importações

Cada volume criado com o `ontap-nas` driver é um FlexVol no cluster do ONTAP. A importação do FlexVols com o `ontap-nas` driver funciona da mesma forma. Um FlexVol que já existe em um cluster ONTAP pode ser importado como `ontap-nas` PVC. Da mesma forma, os vols FlexGroup podem ser importados como `ontap-nas-flexgroup` PVCs.

> Um volume ONTAP deve ser do tipo rw a ser importado pelo Trident. Se um volume for do tipo dp, é um volume de destino SnapMirror; você deve quebrar a relação de espelhamento antes de importar o volume para o Trident.

> O `ontap-nas` driver não pode importar e gerenciar qtrees. Os `ontap-nas` drivers e `ontap-nas-flexgroup` não permitem nomes de volume duplicados.

Por exemplo, para importar um volume nomeado `managed_volume` em um backend `ontap_nas` chamado , use o seguinte comando:

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

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

Para importar um volume chamado `unmanaged_volume` (no `ontap_nas backend`), que o Trident não gerenciará, use o seguinte comando:

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

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

Ao usar o `--no-manage` argumento, o Trident não renomeará o volume nem validará se o volume foi montado. A operação de importação de volume falha se o volume não tiver sido montado manualmente.

> Um bug existente anteriormente com a importação de volumes com UnixPermissions personalizados foi corrigido. Você pode especificar unixPermissions em sua definição de PVC ou configuração de back-end e instruir o Astra Trident a importar o volume de acordo.

#### `ontap-san` importar

O Astra Trident também pode importar ONTAP SAN FlexVols que contenham um único LUN. Isso é consistente com o `ontap-san` driver, que cria um FlexVol para cada PVC e um LUN dentro do FlexVol. Você pode usar o `tridentctl import` comando da mesma forma que em outros casos:

- Inclua o nome `ontap-san` do backend.
- Forneça o nome do FlexVol que precisa ser importado. Lembre-se, este FlexVol contém apenas um LUN

que deve ser importado.

- Fornecer o caminho da definição de PVC que deve ser usado com a `-f` bandeira.
- Escolha entre ter o PVC gerenciado ou não gerenciado. Por padrão, o Trident gerenciará o PVC e renomeará o FlexVol e o LUN no back-end. Para importar como um volume não gerenciado, passe o `--no-manage` sinalizador.

Ao importar um volume não gerenciado `ontap-san`, você deve certificar-se de que o LUN no FlexVol é nomeado `lun0` e é mapeado para um grupo com os iniciadores desejados. O Astra Trident trata isso automaticamente para uma importação gerenciada.

O Astra Trident irá então importar o FlexVol e associá-lo à definição de PVC. O Astra Trident também renomeia o FlexVol para `pvc-<uuid>` o formato e o LUN dentro do FlexVol para `lun0`.

Recomenda-se importar volumes que não tenham conexões ativas existentes. Se você deseja importar um volume usado ativamente, clonar primeiro o volume e, em seguida, fazer a importação.

**Exemplo**

Para importar o `ontap-san-managed` FlexVol que está presente no `ontap_san_default` back-end, execute o `tridentctl import` comando como:

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

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

Um volume ONTAP deve ser do tipo rw para ser importado pelo Astra Trident. Se um volume for do tipo dp, é um volume de destino do SnapMirror; você deve quebrar a relação de espelhamento antes de importar o volume para o Astra Trident.

### `element` importar

É possível importar o software NetApp Element/NetApp HCI volumes para o cluster do Kubernetes com o Trident. Você precisa do nome do seu back-end Astra Trident e do nome exclusivo do volume e do arquivo PVC como argumentos para o `tridentctl import` comando.

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

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

O driver Element suporta nomes de volume duplicados. Se houver nomes de volume duplicados, o processo de importação de volume do Trident retornará um erro. Como solução alternativa, clone o volume e forneça um nome de volume exclusivo. Em seguida, importe o volume clonado.

### `gcp-cvs` importar

Para importar um volume com o suporte do NetApp Cloud Volumes Service no GCP, identifique o volume pelo caminho do volume em vez do nome.

Para importar um `gcp-cvs` volume no back-end chamado `gcpcvs_YEppr` com o caminho de volume `adroit-jolly-swift` do , use o seguinte comando:

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

O caminho do volume é a parte do caminho de exportação do volume após :/. Por exemplo, se o caminho de exportação for `10.0.0.1:/adroit-jolly-swift`, o caminho do volume será `adroit-jolly-swift`.

`azure-netapp-files` **importar**

Para importar um `azure-netapp-files` volume no back-end chamado `azurenetappfiles_40517` com o caminho do volume `importvol1` , execute o seguinte comando:

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

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

O caminho de volume para o volume do ANF está presente no caminho de montagem após :/. Por exemplo, se o caminho de montagem for `10.0.0.2:/importvol1`, o caminho do volume será `importvol1`.

# Prepare o nó de trabalho

Todos os nós de trabalho no cluster do Kubernetes precisam ser capazes de montar os volumes provisionados para os pods. Se você estiver usando o `ontap-nas` driver , `ontap-nas-economy` ou `ontap-nas-flexgroup` para um dos seus backends, os nós de trabalho precisarão das ferramentas NFS. Caso contrário, eles exigem as ferramentas iSCSI.

Versões recentes do RedHat CoreOS têm NFS e iSCSI instalados por padrão.

Você deve sempre reinicializar seus nós de trabalho depois de instalar as ferramentas NFS ou iSCSI, ou então anexar volumes a contentores pode falhar.

## Volumes NFS

| Protocolo | Sistema operacional | Comandos |
|---|---|---|
| NFS | RHEL/CentOS | `sudo yum install -y nfs-utils` |
| NFS | Ubuntu/Debian | `sudo apt-get install -y nfs-common` |

Você deve garantir que o serviço NFS seja iniciado durante o tempo de inicialização.

## Volumes iSCSI

Considere o seguinte ao usar volumes iSCSI:

- Cada nó no cluster do Kubernetes precisa ter uma IQN exclusiva. **Este é um pré-requisito necessário**.
- Se estiver usando RHCOS versão 4,5 ou posterior, RHEL ou CentOS versão 8,2 ou posterior com o `solidfire-san` driver, verifique se o algoritmo de autenticação CHAP está definido como MD5 em `/etc/iscsi/iscsid.conf`.

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

- Ao usar nós de trabalho que executam RHEL/RedHat CoreOS com iSCSI PVs, certifique-se de especificar a `discard` mountOption no StorageClass para executar a recuperação de espaço em linha. "Documentação da RedHat"Consulte .

| Protocolo | Sistema operacional | Comandos |
| --- | --- | --- |
| ISCSI | RHEL/CentOS | 1. Instale os seguintes pacotes de sistema:<br><br>`sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-mapper-multipath`<br><br>2. Verifique se a versão iscsi-iniciador-utils é 6,2.0,874-2.el7 ou posterior:<br><br>`rpm -q iscsi-initiator-utils`<br><br>3. Definir a digitalização para manual:<br><br>`sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/' /etc/iscsi/iscsid.conf`<br><br>4. Ativar multipathing:<br><br>`sudo mpathconf --enable --with_multipathd y --find_multipaths n`<br><br>Certifique-se de `etc/multipath.conf` que contém `find_multipaths no defaults` em .<br><br>5. Certifique-se de que `iscsid` e `multipathd` estão a funcionar:<br><br>`sudo systemctl enable --now iscsid multipathd`<br><br>6. Ativar e iniciar `iscsi`:<br><br>`sudo systemctl enable --now iscsi` |

| Protocolo | Sistema operacional | Comandos |
| --- | --- | --- |
| ISCSI | Ubuntu/Debian | 1. Instale os seguintes pacotes de sistema:<br><br>`sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools scsitools`<br><br>2. Verifique se a versão Open-iscsi é 2,0.874-5ubuntu2.10 ou posterior (para bionic) ou 2,0.874-7.1ubuntu6.1 ou posterior (para focal):<br><br>`dpkg -l open-iscsi`<br><br>3. Definir a digitalização para manual:<br><br>`sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/' /etc/iscsi/iscsid.conf`<br><br>4. Ativar multipathing:<br><br>`sudo tee /etc/multipath.conf <←'EOF'`<br>`defaults {`<br>`user_friendly_names yes`<br>`find_multipaths no`<br>`}`<br>`EOF`<br>`sudo systemctl enable --now multipath-tools.service`<br>`sudo service multipath-tools restart`<br><br>Certifique-se de `etc/multipath.conf` que contém `find_multipaths no` `defaults` em .<br><br>5. Certifique-se de que `open-iscsi` e `multipath-tools` estão ativados e em execução:<br><br>`sudo systemctl status multipath-tools` |

Para o Ubuntu 18,04, você deve descobrir portas de destino com `iscsiadm` antes de iniciar `open-iscsi` o daemon iSCSI para iniciar. Em alternativa, pode modificar o `iscsi` serviço para iniciar `iscsid` automaticamente.

```
sudo systemctl enable
--now open-
iscsi.service
sudo systemctl status
open-iscsi
```

Se você quiser saber mais sobre a preparação automática do nó de trabalho, que é um recurso beta, "aqui"consulte .

# Preparação automática do nó de trabalho

O Astra Trident pode instalar automaticamente as ferramentas e `iSCSI` as necessárias `NFS` nos nós presentes no cluster do Kubernetes. Este é um recurso **beta** e é **não destinado a** clusters de produção. Hoje, o recurso está disponível para nós que executam **CentOS, RHEL e Ubuntu**.

Para esse recurso, o Astra Trident inclui um novo sinalizador de instalação: `--enable-node-prep` Para instalações implantadas com `tridentctl`o . Para implantações com o operador Trident, use a opção Boolean `enableNodePrep`.

A `--enable-node-prep` opção de instalação diz ao Astra Trident para instalar e garantir que os pacotes e/ou serviços NFS e iSCSI estejam sendo executados quando um volume é montado em um nó de trabalho. Este é um recurso **beta** destinado a ser usado em ambientes de desenvolvimento/teste que **não está qualificado** para uso em produção.

Quando o `--enable-node-prep` sinalizador é incluído nas instalações do Astra Trident implantadas com `tridentctl`, veja o que acontece:

1. Como parte da instalação, o Astra Trident Registra os nós em que ele é executado.
2. Quando uma solicitação de reivindicação de volume persistente (PVC) é feita, o Astra Trident cria um PV de um dos back-ends que gerencia.
3. O uso do PVC em um pod exigiria que o Astra Trident montasse o volume no nó em que o pod é executado. O Astra Trident tenta instalar os utilitários de cliente NFS/iSCSI necessários e garantir que os serviços necessários estejam ativos. Isso é feito antes que o volume seja montado.

A preparação de um nó de trabalho é feita apenas uma vez como parte da primeira tentativa feita para montar um volume. Todas as montagens de volume subsequentes devem ser bem-sucedidas desde que nenhuma mudança fora do Astra Trident toque nos `NFS` utilitários e `iSCSI`.

Dessa forma, o Astra Trident pode garantir que todos os nós em um cluster de Kubernetes tenham os utilitários necessários para montar e anexar volumes. Para volumes NFS, a política de exportação também deve permitir que o volume seja montado. O Trident pode gerenciar automaticamente as políticas de exportação por back-end; como alternativa, os usuários podem gerenciar políticas de exportação fora da banda.

# Monitore o Astra Trident

O Astra Trident fornece um conjunto de pontos de extremidade de métricas Prometheus que você pode usar para monitorar a performance do Astra Trident.

As métricas fornecidas pelo Astra Trident permitem que você faça o seguinte:

- Acompanhe a integridade e a configuração do Astra Trident. Você pode examinar como as operações são

bem-sucedidas e se elas podem se comunicar com os backends como esperado.

- Examine as informações de uso do back-end e entenda quantos volumes são provisionados em um back-end e a quantidade de espaço consumido, etc.
- Mantenha um mapeamento da quantidade de volumes provisionados em backends disponíveis.
- Acompanhe o desempenho. Você pode ver quanto tempo leva para que o Astra Trident se comunique com back-ends e realize operações.

> Por padrão, as métricas do Trident são expostas na porta de destino `8001` no `/metrics` endpoint. Essas métricas são **ativadas por padrão** quando o Trident está instalado.

**O que você vai precisar**

- Um cluster Kubernetes com Astra Trident instalado.
- Uma instância Prometheus. Isso pode ser um "Implantação do Prometheus em contêiner" ou você pode optar por executar Prometheus como um "aplicação nativa".

## Passo 1: Defina um alvo Prometheus

Você deve definir um alvo Prometheus para reunir as métricas e obter informações sobre os back-ends que o Astra Trident gerencia, os volumes que ele cria e assim por diante. "blog"Isso explica como você pode usar Prometheus e Grafana com o Astra Trident para recuperar métricas. O blog explica como você pode executar o Prometheus como um operador no cluster Kubernetes e a criação de um ServiceMonitor para obter as métricas do Astra Trident.

## Passo 2: Crie um Prometheus ServiceMonitor

Para consumir as métricas do Trident, você deve criar um Prometheus ServiceMonitor que vigia `trident-csi` o serviço e escuta na `metrics` porta. Um exemplo de ServiceMonitor se parece com isso:

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

Essa definição do ServiceMonitor recupera as métricas retornadas pelo `trident-csi` serviço e procura especificamente o `metrics` ponto final do serviço. Como resultado, prometeu agora está configurado para entender as métricas do Astra Trident.

Além das métricas disponíveis diretamente do Astra Trident, o kubelet expõe muitas `kubelet_volume_*` métricas por meio do seu próprio ponto de extremidade de métricas. O Kubelet pode fornecer informações sobre os volumes anexados e pods e outras operações internas que ele manipula. "aqui"Consulte .

## Passo 3: Consultar métricas do Trident com PromQL

PromQL é bom para criar expressões que retornam dados de séries temporais ou tabulares.

Aqui estão algumas consultas PromQL que você pode usar:

### Obtenha informações de saúde do Trident

- **Porcentagem de respostas HTTP 2XX do Astra Trident**

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

- **Porcentagem de RESPOSTAS REST do Astra Trident via código de status**

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

- **Duração média em ms das operações realizadas pelo Astra Trident**

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

### Obtenha informações de uso do Astra Trident

- **Tamanho médio do volume**

```
trident_volume_allocated_bytes/trident_volume_count
```

- **Espaço total de volume provisionado por cada back-end**

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

#### Obtenha uso de volume individual

Isso é ativado somente se as métricas do kubelet também forem coletadas.

- **Porcentagem de espaço usado para cada volume**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

## Saiba mais sobre a telemetria do Astra Trident AutoSupport

Por padrão, o Astra Trident envia métricas de Prometheus e informações básicas de back-end para o NetApp em uma cadência diária.

- Para impedir que o Astra Trident envie métricas e informações básicas de back-end para o NetApp, passe a `--silence-autosupport` bandeira durante a instalação do Astra Trident.
- O Astra Trident também pode enviar logs de contêiner para o suporte do NetApp sob demanda por meio `tridentctl send autosupport`do . Você precisará acionar o Astra Trident para fazer o upload dos seus logs. Antes de enviar logs, você deve aceitar o NetApp'"política de privacidade"s .
- A menos que especificado, o Astra Trident obtém os logs das últimas 24 horas.
- Você pode especificar o período de retenção do log com o `--since` sinalizador. Por exemplo `tridentctl send autosupport --since=1h`: . Essas informações são coletadas e enviadas por meio `trident-autosupport` de um contêiner que é instalado ao lado do Astra Trident. Pode obter a imagem do contentor em "Trident AutoSupport".
- A Trident AutoSupport não coleta nem transmite informações de identificação pessoal (PII) ou informações pessoais. Ele vem com um "EULA" que não é aplicável à própria imagem de contentor Trident. Você pode saber mais sobre o compromisso da NetApp com a segurança e a confiança dos dados "aqui" .

Um exemplo de payload enviado pelo Astra Trident é parecido com este:

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


- As mensagens do AutoSupport são enviadas para o ponto de extremidade do AutoSupport do NetApp. Se você estiver usando um Registro privado para armazenar imagens de contentor, você pode usar o `--image-registry` sinalizador.
- Você também pode configurar URLs de proxy gerando os arquivos YAML de instalação. Isso pode ser feito usando `tridentctl install --generate-custom-yaml` para criar os arquivos YAML e adicionar o `--proxy-url` argumento para o `trident-autosupport` contentor no `trident-deployment.yaml`.

## Desativar métricas do Astra Trident

Para **desabilitar métricas** de serem reportadas, você deve gerar YAMLs personalizados (usando o `--generate-custom-yaml` sinalizador) e editá-los para remover o `--metrics` sinalizador de ser invocado para o `trident-main` contentor.

## Informações sobre direitos autorais



## Informações sobre marcas comerciais