Usar o Trident

Trident

NetApp
July 01, 2026

Índice

Usar o Trident

Prepare o nó de trabalho

Todos os nós de trabalho no cluster Kubernetes devem ser capazes de montar os volumes que você provisionou para seus pods. Para preparar os nós de trabalho, você deve instalar as ferramentas NFS, iSCSI, NVMe/TCP ou FC, dependendo da seleção do driver.

Selecionando as ferramentas certas

Se você estiver usando uma combinação de drivers, deverá instalar todas as ferramentas necessárias para seus drivers. Versões recentes do Red Hat Enterprise Linux CoreOS (RHCOS) têm as ferramentas instaladas por padrão.

Ferramentas NFS

"Instale as ferramentas NFS" se você estiver usando: `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, ou `azure-netapp-files`.

Ferramentas iSCSI

"Instale as ferramentas iSCSI" se você estiver usando: `ontap-san`, `ontap-san-economy`, `solidfire-san`.

Ferramentas NVMe

"Instale as ferramentas NVMe" se você estiver usando `ontap-san` para nonvolatile memory express (NVMe) sobre o protocolo TCP (NVMe/TCP).

> NetApp recomenda ONTAP 9.12 ou posterior para NVMe/TCP.

Ferramentas SCSI sobre FC

Consulte "Formas de configurar hosts SAN FC e FC-NVMe" para obter mais informações sobre como configurar seus hosts SAN FC e FC-NVMe.

"Instale as ferramentas FC" se você estiver usando `ontap-san` com sanType `fcp` (SCSI sobre FC).

Pontos a considerar: * SCSI sobre FC é suportado em OpenShift e KubeVirt ambientes. * SCSI sobre FC não é suportado no Docker. * A autorrecuperação iSCSI não se aplica ao SCSI sobre FC.

Ferramentas SMB

"Prepare-se para provisionar volumes SMB" se você estiver usando: `ontap-nas` para provisionar volumes SMB.

Descoberta de serviço de nó

Trident tenta detectar automaticamente se o nó pode executar serviços iSCSI ou NFS.

> A descoberta de serviços de nó identifica os serviços descobertos, mas não garante que estejam configurados corretamente. Por outro lado, a ausência de um serviço descoberto não garante que a montagem do volume falhará.

Revisar eventos

Trident cria eventos para o nó a fim de identificar os serviços descobertos. Para revisar esses eventos, execute:

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

Revisar serviços descobertos

Trident identifica os serviços habilitados para cada nó no Trident node CR. Para visualizar os serviços descobertos, execute:

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

Volumes NFS

Instale as ferramentas NFS usando os comandos para o seu sistema operacional. Certifique-se de que o serviço NFS seja iniciado durante a inicialização.

RHEL 8+

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

Ubuntu

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

Reinicie os nós de trabalho após instalar as ferramentas NFS para evitar falhas ao conectar volumes aos contêineres.

volumes iSCSI

Trident pode estabelecer automaticamente uma sessão iSCSI, analisar LUNs, descobrir dispositivos multipath, formatá-los e montá-los em um pod.

Capacidades de autorrecuperação do iSCSI

Para sistemas ONTAP, Trident executa a autorrecuperação iSCSI a cada cinco minutos para:

1. **Identifique** o estado desejado da sessão iSCSI e o estado atual da sessão iSCSI.
2. **Compare** o estado desejado com o estado atual para identificar os reparos necessários. Trident determina as prioridades de reparo e quando antecipar os reparos.
3. **Executar os reparos** necessários para retornar o estado atual da sessão iSCSI ao estado desejado da sessão iSCSI.

Os registros de atividades de autorrecuperação estão localizados no contêiner `trident-main` no respectivo pod do Daemonset. Para visualizar os registros, você deve ter definido `debug` como "true" durante a instalação do Trident.

As capacidades de autorrecuperação do Trident iSCSI podem ajudar a prevenir:

- Sessões iSCSI obsoletas ou com problemas de funcionamento que podem ocorrer após uma falha de conectividade de rede. No caso de uma sessão obsoleta, Trident aguarda sete minutos antes de encerrar a sessão para restabelecer a conexão com um portal.

  Por exemplo, se os segredos CHAP forem rotacionados no controlador de armazenamento e a rede perder a conectividade, os segredos CHAP antigos (desatualizados) podem persistir. A autorrecuperação pode reconhecer isso e restabelecer automaticamente a sessão para aplicar os segredos CHAP atualizados.

- Sessões iSCSI ausentes
- LUNs ausentes

Pontos a considerar antes de atualizar Trident

- Se apenas igroups por nó (introduzidos na 23.04+) estiverem em uso, a autorrecuperação iSCSI iniciará novas varreduras SCSI para todos os dispositivos no barramento SCSI.
- Se apenas igroups com escopo de backend (obsoletos a partir da versão 23.04) estiverem em uso, a autorrecuperação iSCSI iniciará novas varreduras SCSI para os IDs de LUN exatos no barramento SCSI.
- Se uma combinação de igroups por nó e igroups com escopo de backend estiver em uso, a autorrecuperação iSCSI iniciará novas varreduras SCSI para os IDs de LUN exatos no barramento SCSI.

Instale as ferramentas iSCSI

Instale as ferramentas iSCSI usando os comandos para o seu sistema operacional.

Antes de começar

- Cada nó no cluster Kubernetes deve ter um IQN único. **Este é um pré-requisito necessário**.
- Se estiver usando o RHCOS versão 4.5 ou posterior, ou outra distribuição Linux compatível com RHEL, com o `solidfire-san` driver e Element OS 12.5 ou anterior, certifique-se de que o algoritmo de autenticação CHAP esteja definido como MD5 em `/etc/iscsi/iscsid.conf`. Algoritmos CHAP seguros compatíveis com FIPS, SHA1, SHA-256 e SHA3-256 estão disponíveis com Element 12.7.

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

- Ao usar nós de trabalho que executam RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) com iSCSI PVs, especifique o `discard` mountOption em StorageClass para realizar exigência de espaço inline. Consulte "Documentação Red Hat".
- Certifique-se de ter atualizado para a versão mais recente do `multipath-tools`.

RHEL 8+

1. Instale os seguintes pacotes do sistema:

```
sudo yum install -y lsscsi iscsi-initiator-utils device-mapper-
multipath
```

2. Verifique se a versão do iscsi-initiator-utils é 6.2.0.874-2.el7 ou posterior:

```
rpm -q iscsi-initiator-utils
```

3. Defina a digitalização como manual:

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

4. Ative multipathing:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Certifique-se de que `/etc/multipath.conf` contenha `find_multipaths no` sob `defaults`.

5. Certifique-se de que `iscsid` e `multipathd` estejam em execução:

```
sudo systemctl enable --now iscsid multipathd
```

6. Ativar e iniciar `iscsi`:

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. Instale os seguintes pacotes do sistema:

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. Verifique se a versão do open-iscsi é 2.0.874-5ubuntu2.10 ou posterior (para bionic) ou 2.0.874-7.1ubuntu6.1 ou posterior (para focal):

```
dpkg -l open-iscsi
```

3. Defina a digitalização como manual:

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

4. Ative multipathing:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> Certifique-se de que `/etc/multipath.conf` contenha `find_multipaths no` sob `defaults`.

5. Certifique-se de que `open-iscsi` e `multipath-tools` estejam habilitados e em execução:

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

> Para o Ubuntu 18.04, você deve descobrir as portas de destino com `iscsiadm` antes de iniciar `open-iscsi` para que o daemon iSCSI seja iniciado. Você pode, alternativamente, modificar o serviço `iscsi` para iniciar `iscsid` automaticamente.

Configurar ou desativar autorrecuperação iSCSI

Você pode configurar as seguintes configurações de autorrecuperação do Trident iSCSI para corrigir sessões obsoletas:

- **Intervalo de autorrecuperação do iSCSI**: Determina a frequência com que a autorrecuperação do iSCSI é acionada (padrão: 5 minutos). Você pode configurá-lo para ser executado com mais frequência definindo um número menor ou com menos frequência definindo um número maior.

Definir o intervalo de autorrecuperação do iSCSI para 0 interrompe a autorrecuperação do iSCSI completamente. Não recomendamos desativar a autorrecuperação do iSCSI; ela só deve ser desativada em certos cenários, quando a autorrecuperação do iSCSI não estiver funcionando conforme o esperado ou para fins de depuração.

- **iSCSI Self-Healing Wait Time**: Determina a duração que a autorrecuperação do iSCSI aguarda antes de fazer logout de uma sessão com problemas e tentar fazer login novamente (padrão: 7 minutos). Você pode configurar para um número maior, de modo que as sessões identificadas como com problemas tenham que esperar mais tempo antes de serem desconectadas e, em seguida, uma tentativa de login seja feita, ou para um número menor para fazer logout e login mais cedo.

Helm

Para configurar ou alterar as configurações de autorrecuperação do iSCSI, passe os `iscsiSelfHealingInterval` e `iscsiSelfHealingWaitTime` parâmetros durante a instalação ou atualização do helm.

O exemplo a seguir define o intervalo de autorrecuperação do iSCSI para 3 minutos e o tempo de espera para autorrecuperação para 6 minutos:

```
helm install trident trident-operator-100.2602.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

tridentctl

Para configurar ou alterar as configurações de autorrecuperação do iSCSI, passe os `iscsi-self-healing-interval` e `iscsi-self-healing-wait-time` parâmetros durante a instalação ou atualização do tridentctl.

O exemplo a seguir define o intervalo de autorrecuperação do iSCSI para 3 minutos e o tempo de espera para autorrecuperação para 6 minutos:

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

Volumes NVMe/TCP

Instale as ferramentas NVMe usando os comandos para o seu sistema operacional.

- NVMe requer RHEL 9 ou posterior.
- Se a versão do kernel do seu nó Kubernetes for muito antiga ou se o pacote NVMe não estiver disponível para a sua versão do kernel, talvez seja necessário atualizar a versão do kernel do seu nó para uma com o pacote NVMe.

RHEL 9

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

Ubuntu

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

Verificar instalação

Após a instalação, verifique se cada nó no cluster Kubernetes possui um NQN exclusivo usando o comando:

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

Trident modifica o `ctrl_device_tmo` valor para garantir que o NVMe não abandone o caminho caso ocorra uma falha. Não altere essa configuração.

Volumes SCSI over FC

Agora você pode usar o protocolo Fibre Channel (FC) com Trident para provisionar e gerenciar recursos de storage no sistema ONTAP.

Pré-requisitos

Configurar as definições de rede e de nó necessárias para FC.

Configurações de rede

1. Obtenha o WWPN das interfaces de destino. Consulte "network interface show" para mais informações.
2. Obtenha o WWPN para as interfaces no iniciador (Host).

   Consulte os utilitários correspondentes do sistema operacional do host.

3. Configurar o zoneamento no switch FC usando os WWPNs do host e do destino.

   Consulte a documentação do respectivo fornecedor do switch para obter informações.

   Consulte a seguinte documentação do ONTAP para obter detalhes:

   - "Visão geral do zoneamento Fibre Channel e FCoE"
   - "Formas de configurar hosts SAN FC e FC-NVMe"

Instale as ferramentas FC

Instale as ferramentas FC usando os comandos para o seu sistema operacional.

- Ao usar nós de trabalho que executam RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) com FC PVs, especifique o `discard` mountOption na StorageClass para realizar exigência de espaço inline. Consulte "Documentação Red Hat".

RHEL 8+

1. Instale os seguintes pacotes do sistema:

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. Ative multipathing:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Certifique-se de que `/etc/multipath.conf` contenha `find_multipaths no` sob `defaults`.

3. Certifique-se de que `multipathd` está em execução:

```
sudo systemctl enable --now multipathd
```

Ubuntu

1. Instale os seguintes pacotes do sistema:

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. Ative multipathing:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> Certifique-se de que `/etc/multipath.conf` contenha `find_multipaths no` sob `defaults`.

3. Certifique-se de que `multipath-tools` está ativado e em execução:

```
sudo systemctl status multipath-tools
```

Prepare-se para provisionar volumes SMB

Você pode provisionar volumes SMB usando `ontap-nas` drivers.

> Você deve configurar ambos os protocolos NFS e SMB/CIFS na SVM para criar um `ontap-nas-economy` volume SMB para clusters ONTAP locais. A falha ao configurar qualquer um desses protocolos fará com que a criação do volume SMB falhe.

> `autoExportPolicy` não é compatível com volumes SMB.

Antes de começar

Antes de poder provisionar volumes SMB, você deve ter o seguinte.

- Um cluster Kubernetes com um nó controlador Linux e pelo menos um nó de trabalho Windows executando Windows Server 2022. Trident suporta volumes SMB montados em pods executados apenas em nós Windows.
- Pelo menos um segredo Trident contendo suas credenciais do Active Directory. Para gerar o segredo `smbcreds`:

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

- Um proxy CSI configurado como um serviço do Windows. Para configurar um `csi-proxy`, consulte "GitHub: CSI Proxy" ou "GitHub: CSI Proxy para Windows" para nós do Kubernetes em execução no Windows.

Passos

1. Para o ONTAP local, você pode opcionalmente criar um compartilhamento SMB ou o Trident pode criar um para você.

   > Os compartilhamentos SMB são necessários para Amazon FSx for ONTAP.

   Você pode criar os compartilhamentos administrativos SMB de duas maneiras: usando o "Microsoft Management Console" snap-in Shared Folders ou usando a ONTAP CLI. Para criar os compartilhamentos SMB usando a ONTAP CLI:

   a. Se necessário, crie a estrutura de caminho de diretórios para o compartilhamento.

      O `vserver cifs share create` comando verifica o caminho especificado na opção -path durante a criação do compartilhamento. Se o caminho especificado não existir, o comando falha.

   b. Crie um compartilhamento SMB associado à SVM especificada:

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

c. Verifique se o compartilhamento foi criado:

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

Consulte "Criar um compartilhamento SMB" para obter detalhes completos.

2. Ao criar o backend, você deve configurar o seguinte para especificar os volumes SMB. Para todas as opções de configuração do backend do FSx para ONTAP, consulte "Opções e exemplos de configuração do FSx for ONTAP".

Parâmetro	Descrição	Exemplo
`smbShare`	Você pode especificar uma das seguintes opções: o nome de um compartilhamento SMB criado usando o Microsoft Management Console ou ONTAP CLI; um nome para permitir que o Trident crie o compartilhamento SMB; ou você pode deixar o parâmetro em branco para impedir o acesso comum aos volumes. Este parâmetro é opcional para ONTAP on-premises. Este parâmetro é obrigatório para Amazon FSx for ONTAP backends e não pode ficar em branco.	`smb-share`
`nasType`	**Deve ser definido como `smb`.** Se for nulo, o padrão é `nfs`.	`smb`
`securityStyle`	Estilo de segurança para novos volumes. **Deve ser definido como `ntfs` ou `mixed` para volumes SMB.**	`ntfs` ou `mixed` for SMB volumes
`unixPermissions`	Modo para novos volumes. **Deve ser deixado em branco para volumes SMB.**	""

Configurar e gerenciar backends

Configurar backends

Um backend define a relação entre Trident e um sistema de storage. Ele informa ao Trident como se comunicar com esse sistema de storage e como o Trident deve provisionar volumes a partir dele.

Trident oferece automaticamente pools de armazenamento de backends que correspondem aos requisitos definidos por uma classe de armazenamento. Saiba como configurar o backend para o seu sistema de storage.

- "Configurar um backend do Azure NetApp Files"
- "Configurar um backend do Google Cloud NetApp Volumes"
- "Configurar um NetApp HCI ou SolidFire backend"
- "Configure um backend com drivers NAS do ONTAP ou Cloud Volumes ONTAP"

- "Configure um backend com drivers SAN do ONTAP ou Cloud Volumes ONTAP"
- "Use Trident com Amazon FSx for NetApp ONTAP"

Azure NetApp Files

Configurar um backend do Azure NetApp Files

Use o Azure NetApp Files como backend para Trident. Esse backend oferece suporte a volumes NFS e SMB. Trident oferece suporte a identidades gerenciadas e identidade de carga de trabalho para clusters do Azure Kubernetes Service (AKS).

Ambientes de nuvem Azure suportados

Trident oferece suporte a back-ends do Azure NetApp Files em vários ambientes de nuvem do Azure.

As nuvens do Microsoft Azure compatíveis incluem:

- Azure Commercial
- Azure Government (Azure Government / MAG)

Ao implantar Trident ou configurar um backend do Azure NetApp Files, certifique-se de que o Azure Resource Manager e os endpoints de autenticação correspondam ao seu ambiente de nuvem do Azure.

Revise o suporte do driver do Azure NetApp Files

Trident fornece o seguinte driver de armazenamento Azure NetApp Files.

Os modos de acesso suportados incluem *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX) e *ReadWriteOncePod* (RWOP).

Driver	Protocolo	volumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`azure-netapp-files`	NFS SMB	Sistema de arquivos	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Revisar considerações

- Azure NetApp Files não suporta volumes menores que 50 GiB. Trident cria um volume de 50 GiB quando um volume menor é solicitado.
- Trident suporta volumes SMB montados em pods executados apenas em nós Windows.
- As implantações do Azure NetApp Files em nuvens Azure não comerciais exigem endpoints do Azure Resource Manager e de autenticação específicos da nuvem. Certifique-se de que Trident e qualquer configuração de backend usem os endpoints apropriados para o seu ambiente de nuvem Azure.

Use identidades gerenciadas para AKS

Trident oferece suporte "identidades gerenciadas" para clusters AKS.

Se você utiliza `tridentctl` para criar ou gerenciar back-ends do Azure NetApp Files, certifique-se de que ele esteja configurado para o ambiente de nuvem Azure correto.

Para usar identidades gerenciadas, você deve ter:

- Um cluster Kubernetes implantado usando AKS
- Identidades gerenciadas configuradas no cluster Kubernetes do AKS
- Trident instalado com `cloudProvider` definido para `"Azure"`

Operador Trident

Edite `tridentorchestrator_cr.yaml` e defina `cloudProvider` como `"Azure"`.

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

Helm

O exemplo a seguir instala Trident e define `cloudProvider` usando a variável de ambiente `$CP`:

```
helm install trident trident-operator-100.2602.0.tgz --create-namespace
--namespace <trident-namespace> --set cloudProvider=$CP
```

<code>tridentctl</code>

O exemplo a seguir instala Trident e define o `cloud-provider` sinalizador para `Azure`:

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

Use identidade de carga de trabalho para AKS

A identidade de carga de trabalho permite que os pods do Kubernetes acessem recursos do Azure autenticando-se como uma identidade de carga de trabalho.

Se você utiliza `tridentctl` para criar ou gerenciar back-ends do Azure NetApp Files, certifique-se de que ele esteja configurado para o ambiente de nuvem Azure correto.

Para usar workload identity, você precisa ter:

- Um cluster Kubernetes implantado usando AKS
- Identidade de carga de trabalho e oidc-issuer configurados no cluster Kubernetes do AKS

- Trident instalado com `cloudProvider` definido para `"Azure"` e `cloudIdentity` definido para o valor de identidade da carga de trabalho

Operador Trident

Edite `tridentorchestrator_cr.yaml` e defina `cloudProvider` como `"Azure"`. Defina `cloudIdentity` como `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx' # Edit
```

Helm

Defina os valores para os parâmetros **cloud-provider (CP)** e **cloud-identity (CI)** usando as seguintes variáveis de ambiente:

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx'"
```

O exemplo a seguir instala Trident e define `cloudProvider` usando `$CP` e define `cloudIdentity` usando `$CI`:

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

<code>tridentctl</code>

Defina os valores para os parâmetros **cloud provider** e **cloud identity** usando as seguintes variáveis de ambiente:

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx"
```

O exemplo a seguir instala Trident e define `cloud-provider` como `$CP` e `cloud-identity` como `$CI`:

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

Prepare-se para configurar um backend do Azure NetApp Files

Antes de configurar seu backend do Azure NetApp Files, você precisa garantir que os seguintes requisitos sejam atendidos.

Ambientes de nuvem Azure suportados

Trident oferece suporte a back-ends do Azure NetApp Files em vários ambientes de nuvem do Azure.

As nuvens do Microsoft Azure compatíveis incluem:

- Azure Commercial
- Azure Government (Azure Government / MAG)

Ao preparar seu ambiente, certifique-se de que sua assinatura do Azure, configuração de identidade e recursos do Azure NetApp Files sejam criados no ambiente de nuvem do Azure apropriado.

Pré-requisitos para volumes NFS e SMB

Se você estiver usando o Azure NetApp Files pela primeira vez ou em um novo local, será necessária alguma configuração inicial para configurar o Azure NetApp Files e criar um volume NFS. Consulte "Azure: configure o Azure NetApp Files e crie um volume NFS".

Para configurar e usar um "Azure NetApp Files" backend, você precisa do seguinte:

> - `subscriptionID`, `tenantID`, `clientID`, `location` e `clientSecret` são opcionais ao usar identidades gerenciadas em um cluster AKS.
> - `tenantID`, `clientID`, e `clientSecret` são opcionais ao usar uma identidade de nuvem em um cluster AKS.
> - As implantações do Azure NetApp Files em nuvens Azure não comerciais exigem endpoints do Azure Resource Manager e de autenticação específicos da nuvem. Certifique-se de que Trident e qualquer configuração de backend usem os endpoints apropriados para o seu ambiente de nuvem Azure.

- Um pool de capacidade. Consulte "Microsoft: criar um pool de capacidade para Azure NetApp Files".
- Uma sub-rede delegada ao Azure NetApp Files. Consulte "Microsoft: delegar uma sub-rede ao Azure NetApp Files".
- `subscriptionID` de uma assinatura do Azure com o Azure NetApp Files ativado.
- `tenantID`, `clientID`, e `clientSecret` de um "Registro de aplicativo" no Azure Active Directory com permissões suficientes para o serviço NetApp Files do Azure. O registro do aplicativo deve usar um dos seguintes:
  - O papel de Owner ou Contributor "predefinido pelo Azure".
  - Uma "Função de Contributor personalizada" no nível da assinatura (`assignableScopes` com as seguintes permissões, que são limitadas apenas ao que Trident exige. Após criar a função personalizada, "atribua a função usando o portal do Azure".

Função de colaborador personalizado

```
{
  "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.NetApp/netAppAccounts/capacityPools/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/delete",
```

```
            "Microsoft.Features/features/read",
            "Microsoft.Features/operations/read",
            "Microsoft.Features/providers/features/read",

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

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

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

- 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 do backend. Os valores de localização especificados em pools virtuais são ignorados.
- Para usar `Cloud Identity`, obtenha o `client ID` de um "identidade gerenciada atribuída pelo usuário" e especifique esse ID em `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

Requisitos adicionais para volumes SMB

Para criar um volume SMB, você deve ter:

- Active Directory configurado e conectado ao Azure NetApp Files. Consulte "Microsoft: criar e gerenciar conexões do Active Directory para Azure NetApp Files".
- Um cluster Kubernetes com um nó controlador Linux e pelo menos um nó de trabalho Windows executando Windows Server 2022. Trident suporta volumes SMB montados em pods executados apenas em nós Windows.
- Pelo menos um segredo do Trident contendo suas credenciais do Active Directory para que o Azure NetApp Files possa autenticar no Active Directory. Para gerar o segredo `smbcreds`:

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

- Um proxy CSI configurado como um serviço do Windows. Para configurar um `csi-proxy`, consulte "GitHub: CSI Proxy" ou "GitHub: CSI Proxy para Windows" para nós do Kubernetes em execução no Windows.

Opções e exemplos de configuração do backend do Azure NetApp Files

Saiba mais sobre as opções de configuração de back-end NFS e SMB para o Azure NetApp Files e veja exemplos de configuração.

Opções de configuração do backend

Trident usa sua configuração de back-end (sub-rede, rede virtual, nível de serviço e localização) para criar volumes do Azure NetApp Files em pools de capacidade disponíveis na localização solicitada e que correspondam ao nível de serviço e à sub-rede solicitados.

Os backends do Azure NetApp Files fornecem essas opções de configuração.

Parâmetro	Descrição	Padrão
`version`	Versão de configuração do backend.	Sempre 1
`storageDriverName`	Nome do driver de armazenamento	"azure-netapp-files"
`backendName`	Nome personalizado para o backend de armazenamento	Nome do driver + "_" + caracteres aleatórios
`subscriptionID`	O ID da assinatura da sua assinatura do Azure opcional quando as identidades gerenciadas estão habilitadas em um cluster do AKS.	
`tenantID`	O ID do locatário de um registro de aplicativo opcional quando identidades gerenciadas ou identidade na nuvem são usadas em um cluster AKS.	
`clientID`	O ID do cliente de um registro de aplicativo é opcional quando identidades gerenciadas ou identidade na nuvem são usadas em um cluster AKS.	
`clientSecret`	O segredo do cliente de um registro de aplicativo. Opcional quando identidades gerenciadas ou identidade na nuvem são usadas em um cluster AKS.	
`serviceLevel`	Um de `Standard`, `Premium` ou `Ultra`	"" (aleatório)
`location`	Nome da localização do Azure onde os novos volumes serão criados Opcional quando as identidades gerenciadas estão habilitadas em um cluster AKS.	
`resourceGroups`	Lista de grupos de recursos para filtrar recursos descobertos	"[]" (sem filtro)

Parâmetro	Descrição	Padrão
`netappAccounts`	Lista de contas NetApp para filtrar 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 a `Microsoft.Netapp/volumes`	""
`networkFeatures`	Conjunto de recursos de VNet para um volume, pode ser `Basic` ou `Standard`. Network Features não está disponível em todas as regiões e pode precisar ser habilitado em uma assinatura. Especificar `networkFeatures` quando a funcionalidade não está habilitada faz com que o provisionamento do volume falhe.	""
`nfsMountOptions`	Controle preciso das opções de montagem NFS. Ignorado para volumes SMB. Para montar volumes usando a versão de NFS 4.1, inclua `nfsvers=4` na lista de opções de montagem separadas por vírgula para escolher NFS v4.1. As opções de montagem definidas em uma definição de classe de armazenamento substituem as opções de montagem definidas na configuração do backend.	"nfsvers=3"
`limitVolumeSize`	Falhar no provisionamento se o tamanho do volume solicitado for superior a este valor	"" (não aplicado por padrão)
`debugTraceFlags`	Sinalizadores de depuração para usar na resolução de problemas. Exemplo, `\{"api": false, "method": true, "discovery": true}`. Não use isso a menos que esteja solucionando problemas e precise de um despejo de log detalhado.	null
`nasType`	Configurar a criação de volumes NFS ou SMB. As opções são `nfs`, `smb` ou null. Definir como null define volumes NFS por padrão.	`nfs`

Parâmetro	Descrição	Padrão
`supportedTopologies`	Representa uma lista de regiões e zonas suportadas por este backend. Para obter mais informações, consulte "Usar a topologia CSI".	
`qosType`	Representa o tipo de QoS: automático ou manual.	Automático
`maxThroughput`	Define a taxa de transferência máxima permitida em MiB/s. Compatível apenas com pools de capacidade QoS manuais.	`4 MiB/sec`

Para mais informações sobre recursos de rede, consulte "Configurar recursos de rede para um volume do Azure NetApp Files".

Considere ambientes de nuvem Azure (26.02)

A partir da versão 26.02, Trident oferece suporte à criação e ao gerenciamento de back-ends do Azure NetApp Files em vários ambientes de nuvem do Azure.

As nuvens do Microsoft Azure compatíveis incluem:

- Azure Commercial
- Azure Government (Azure Government / MAG)

Ao implantar Trident ou criar um backend do Azure NetApp Files, certifique-se de que os endpoints do Azure Resource Manager e de autenticação correspondam ao seu ambiente de nuvem do Azure. Se os endpoints não corresponderem, `tridentctl` não será possível autenticar e a criação do backend falhará.

Permissões e recursos necessários

Se você receber um erro "Nenhum pool de capacidade encontrado" ao criar um PVC, é provável que o registro do seu aplicativo não tenha as permissões e os recursos necessários (sub-rede, rede virtual, pool de capacidade) associados. Se o modo de depuração estiver habilitado, Trident registra os recursos do Azure descobertos quando o backend é criado. Verifique se uma função apropriada está sendo usada.

Os valores para `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork` e `subnet` podem ser especificados usando nomes curtos ou totalmente qualificados. Nomes totalmente qualificados são recomendados na maioria das situações, pois nomes curtos podem corresponder a vários recursos com o mesmo nome.

Se a vNet estiver localizada em um grupo de recursos diferente da conta de armazenamento Azure NetApp Files (ANF), especifique o grupo de recursos para a rede virtual ao configurar a lista resourceGroups para o backend.

Os `resourceGroups`, `netappAccounts` e `capacityPools` valores são filtros que restringem o conjunto de recursos descobertos àqueles disponíveis para este backend de armazenamento e podem ser especificados em qualquer combinação. Os nomes totalmente qualificados seguem este formato:

Tipo	Formatar
Grupo de recursos	<resource group>
Conta do NetApp	<resource group>/<netapp account>
Pool de capacidade	<resource group>/<netapp account>/<capacity pool>
Rede virtual	<resource group>/<virtual network>
Sub-rede	<resource group>/<virtual network>/<subnet>

Provisionamento de volume

Você pode controlar o provisionamento de volumes padrão especificando as seguintes opções em uma seção especial do arquivo de configuração. Consulte Exemplos de configurações para obter detalhes.

Parâmetro	Descrição	Padrão
`exportRule`	Regras de exportação para novos volumes. `exportRule` deve ser uma lista separada por vírgulas de qualquer combinação de endereços IPv4 ou sub-redes IPv4 na notação CIDR. Ignorado para volumes SMB.	"0.0.0.0/0"
`snapshotDir`	Acesso ao `.snapshot` diretório	`true`, `false` (Definido explicitamente).
`size`	O tamanho padrão de novos volumes	"100G"
`unixPermissions`	As permissões unix de novos volumes (4 dígitos octais). Ignorado para volumes SMB.	"" (recurso em pré-visualização, requer inclusão na lista de permissões na assinatura)

Exemplos de configurações

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

Configuração mínima

Esta é a configuração mínima absoluta de backend. Com esta configuração, Trident descobre todas as suas NetApp contas, pools de capacidade e sub-redes delegadas ao Azure NetApp Files no local configurado e coloca novos volumes em um desses pools e sub-redes aleatoriamente. Como `nasType` foi omitido, o `nfs` padrão se aplica e o backend irá provisionar volumes NFS.

Essa configuração é ideal quando você está começando a usar o Azure NetApp Files e testando as funcionalidades, mas, na prática, você vai querer fornecer um escopo adicional para os volumes que você provisionar.

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  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
```

Identidades gerenciadas para AKS

Esta configuração de backend omite `subscriptionID`, `tenantID`, `clientID` e `clientSecret`, que são opcionais ao usar identidades gerenciadas.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - resource-group-1/netapp-account-1/ultra-pool
  resourceGroups:
    - resource-group-1
  netappAccounts:
    - resource-group-1/netapp-account-1
  virtualNetwork: resource-group-1/eastus-prod-vnet
  subnet: resource-group-1/eastus-prod-vnet/eastus-anf-subnet
```

Identidade na nuvem para AKS

Esta configuração de backend omite `tenantID`, `clientID` e `clientSecret`, que são opcionais ao usar uma identidade na nuvem.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

Configuração específica de nível de serviço com filtros de pool de capacidade

Essa configuração de backend coloca volumes na localização do Azure `eastus` em um `Ultra` pool de capacidade. Trident descobre automaticamente todas as sub-redes delegadas ao Azure NetApp Files 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 de backend com pools de capacidade QoS manuais

Essa configuração de backend coloca volumes na localização do Azure `eastus` com pools de capacidade de QoS manuais.

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

Configuração avançada

Essa configuração de backend reduz ainda mais o escopo do posicionamento de volumes para uma única sub-rede e também modifica alguns padrões de provisionamento de volumes.

```
---
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: application-group-1/eastus-prod-vnet
subnet: application-group-1/eastus-prod-vnet/my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: "true"
  size: 200Gi
  unixPermissions: "0777"
```

Configuração de pool virtual

Esta configuração de backend define vários pools de armazenamento em um único arquivo. Isso é útil quando você tem vários pools de capacidade que suportam diferentes níveis de serviço e deseja criar classes de armazenamento no Kubernetes que os representem. Rótulos de pool virtual foram usados para diferenciar os pools com base em `performance`.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
  - application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
  - labels:
      performance: gold
    serviceLevel: Ultra
    capacityPools:
      - application-group-1/netapp-account-1/ultra-1
      - application-group-1/netapp-account-1/ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - application-group-1/netapp-account-1/premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - application-group-1/netapp-account-1/standard-1
      - application-group-1/netapp-account-1/standard-2
```

Configuração de topologias suportadas

Trident facilita o provisionamento de volumes para cargas de trabalho com base em regiões e zonas de disponibilidade. O `supportedTopologies` bloco nesta configuração de backend é usado para fornecer uma lista de regiões e zonas por backend. Os valores de região e zona especificados aqui devem corresponder aos valores de região e zona dos rótulos em cada nó de cluster Kubernetes. Essas regiões e zonas representam a lista de valores permitidos que podem ser fornecidos em uma classe de armazenamento. Para classes de armazenamento que contêm um subconjunto das regiões e zonas fornecidas em um backend, Trident cria volumes na região e zona mencionadas. Para obter mais informações, consulte "Usar a topologia CSI".

```
---
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
supportedTopologies:
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-1
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-2
```

Definições de classe de armazenamento

As seguintes `StorageClass`definições referem-se aos pools de armazenamento acima.

Exemplo de definições usando `parameter.selector` field

Usando `parameter.selector` você pode especificar para cada `StorageClass` o pool virtual que é usado para hospedar um volume. O volume terá os aspectos 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
```

Exemplos de definições para volumes SMB

Usando `nasType`, `node-stage-secret-name` e `node-stage-secret-namespace`, você pode especificar um volume SMB e fornecer as credenciais necessárias do Active Directory.

Configuração básica no namespace padrão

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

Usando segredos diferentes por namespace

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

Utilizando segredos diferentes em cada volume

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

`nasType: smb`Filtros para pools que suportam volumes SMB. `nasType: nfs` ou `nasType: null`filtros para pools NFS.

Criar o backend

Após criar o arquivo de configuração de backend, execute o seguinte comando:

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

Se você usa uma nuvem Azure não comercial, certifique-se de que `tridentctl` está configurado para usar o Azure Resource Manager e os endpoints de autenticação para o seu ambiente de nuvem Azure. Se a criação do backend falhar, verifique a configuração do backend e visualize os logs para determinar a causa:

```
tridentctl logs
```

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

Google Cloud NetApp Volumes

Configurar Google Cloud NetApp Volumes

Você pode configurar o Google Cloud NetApp Volumes como um backend para Trident a fim de provisionar storage para cargas de trabalho do Kubernetes.

Visão geral

Trident oferece suporte ao Google Cloud NetApp Volumes para cargas de trabalho NAS (NFS e SMB) e em bloco (iSCSI).

- As cargas de trabalho NAS usam o `google-cloud-netapp-volumes` backend
- As cargas de trabalho em bloco (iSCSI) usam o `google-cloud-netapp-volumes-san`backend

Os volumes NAS fornecem armazenamento baseado em arquivos e são acessados usando os protocolos NFS ou SMB. Esses volumes suportam acesso compartilhado entre vários pods ou nós.

Os volumes de bloco fornecem armazenamento bruto em bloco e são acessados como dispositivos iSCSI conectados a nós do Kubernetes. Esses volumes são usados quando os aplicativos exigem acesso em nível de bloco.

Isso se aplica aos seguintes ambientes:

- Trident 26.02 e posteriores
- Google Kubernetes Engine (GKE) ou Red Hat OpenShift
- Pools de armazenamento do Google Cloud NetApp Volumes

Para configurar o armazenamento em bloco (iSCSI), consulte "Configurar armazenamento em bloco (iSCSI)".

Prepare-se para configurar

A identidade na nuvem permite que as cargas de trabalho do Kubernetes acessem recursos do Google Cloud autenticando-se como uma identidade de carga de trabalho em vez de usar credenciais estáticas.

Para usar a identidade na nuvem com Google Cloud NetApp Volumes, você deve ter:

- Um cluster Kubernetes implantado usando Google Kubernetes Engine (GKE)
- Identidade de carga de trabalho habilitada no cluster GKE e o servidor de metadados habilitado nos pools de nós
- Uma conta de serviço do Google Cloud com a função de Administrador do Google Cloud NetApp Volumes (`roles/netapp.admin`) ou uma função personalizada equivalente
- Trident instalado com o provedor de nuvem definido como `GCP` e a anotação de identidade da nuvem configurada

Operador Trident

Para instalar o Trident usando o operador Trident, edite `tridentorchestrator_cr.yaml`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  cloudProvider: "GCP"
  cloudIdentity: "iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com"
```

Helm

Defina o provedor de nuvem e a identidade da nuvem ao instalar o Trident com o Helm:

```
helm install trident trident-operator-100.6.0.tgz \
  --set cloudProvider=GCP \
  --set cloudIdentity="iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com"
```

tridentctl

Instale Trident especificando o provedor de nuvem e a identidade de nuvem:

```
tridentctl install \
  --cloud-provider=GCP \
  --cloud-identity="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com" \
  -n trident
```

Configurar armazenamento NAS

Para pools de armazenamento UNIFIED do Google Cloud NetApp Volumes, Trident aplica regras de nomenclatura e validação específicas do UNIFIED durante as operações de volume.

Ao localizar um volume, Trident pode avaliar várias variantes de nomes de volume compatíveis (por exemplo, formatos com hífen e sublinhado) para melhorar a confiabilidade da importação e da descoberta.

Detalhes do driver

Trident fornece o `google-cloud-netapp-volumes` driver para provisionar armazenamento NAS a partir do Google Cloud NetApp Volumes.

O driver suporta os seguintes modos de acesso:

- ReadWriteOnce (RWO)
- ReadOnlyMany (ROX)
- ReadWriteMany (RWX)
- ReadWriteOncePod (RWOP)

Driver	Protocolo	volumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`google-cloud-netapp-volumes`	NFS SMB	Sistema de arquivos	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Configurar um backend NAS do Trident

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
```

Provisionar volumes NAS

Os volumes NAS são provisionados usando o `google-cloud-netapp-volumes` backend e suportam os protocolos NFS e SMB.

StorageClass for volumes NFS

Para provisionar volumes NFS, defina `nasType` para `nfs`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "nfs"
allowVolumeExpansion: true
```

StorageClass for volumes SMB

Para provisionar volumes SMB, defina `nasType` para `smb` e forneça as credenciais.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
allowVolumeExpansion: true
```

Exemplo de PersistentVolumeClaim (RWX)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwx
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

Exemplo de PersistentVolumeClaim (RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

Volumes NAS usam `volumeMode: Filesystem`.

Configurar o Google Cloud NetApp Volumes para cargas de trabalho SAN

Você pode configurar Trident para provisionar volumes de armazenamento em bloco usando o protocolo iSCSI do Google Cloud NetApp Volumes. Os volumes SAN são provisionados a partir de pools de armazenamento Flex Unified usando o `google-cloud-netapp-volumes-san` driver de armazenamento.

Este driver é dedicado a cargas de trabalho em bloco e não oferece suporte a protocolos NAS.

O `google-cloud-netapp-volumes-san`backend é necessário para provisionar volumes de bloco iSCSI. O `google-cloud-netapp-volumes`backend suporta apenas protocolos NAS e não pode ser usado para cargas de trabalho SAN.

Visão geral

Trident oferece suporte ao Google Cloud NetApp Volumes SAN (iSCSI) para cargas de trabalho usando o driver `google-cloud-netapp-volumes-san`.

Os volumes SAN são provisionados a partir de pools de armazenamento Flex Unified e apresentados aos nós do Kubernetes como dispositivos de bloco iSCSI.

Isso se aplica aos seguintes ambientes:

- Trident 26.02 e posteriores
- Google Kubernetes Engine (GKE) ou Red Hat OpenShift
- Pools de storage unificado do Google Cloud NetApp Volumes Flex
- Cargas de trabalho baseadas em iSCSI

Pools de storage unificado Flex

Os pools de armazenamento Flex Unified fornecem armazenamento em bloco usando o protocolo iSCSI e são necessários para o provisionamento de SAN:

- Os pools regionais Flex Unified são suportados.
- Os pools Zonais unificados Flex são suportados a partir do Trident 26.02.1.
- Apenas o nível de serviço **Flex** é compatível com cargas de trabalho SAN.

Configurar um backend SAN do Trident

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-san
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
      performance: flex
    network: "<vpc-network>"
    serviceLevel: Flex
```

Crie um StorageClass

Após configurar o backend SAN, crie um StorageClass que faça referência ao `google-cloud-netapp-volumes-san` driver.

O tipo de sistema de arquivos é definido no StorageClass, não no backend.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Tipos de sistemas de arquivos suportados:

- `ext4` (padrão)
- `ext3`

- `xfs`

> O driver SAN suporta apenas o nível de serviço Flex e não utiliza parâmetros de backend específicos do NAS, como `exportRule`, `unixPermissions`, `nasType`, `snapshotDir`, `nfsMountOptions` ou configurações relacionadas ao tiering.

Provisionar volumes de bloco

ReadWriteOnce (RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadWriteOncePod (RWOP)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwop
spec:
  accessModes:
    - ReadWriteOncePod
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadOnlyMany (ROX)

Um padrão comum para o ROX é clonar um volume ReadWriteOnce existente e montar o clone como somente leitura.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rox
spec:
  accessModes:
    - ReadOnlyMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
  dataSource:
    kind: PersistentVolumeClaim
    name: gcnv-san-rwo
```

ReadWriteMany (RWX) — somente bloco bruto

ReadWriteMany é suportado somente quando `volumeMode: Block`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-raw-rwx
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

Comportamento do volume do bloco

Os volumes de bloco são provisionados como LUNs iSCSI e apresentados aos nós do Kubernetes como dispositivos de bloco.

Volumes de bloco:

- Use o protocolo iSCSI
- Suporte ao sistema de arquivos e apresentação de blocos brutos
- Estão anexados e gerenciados pela Trident
- Suporte a múltiplos modos de acesso do Kubernetes

Modos de acesso

Os volumes de bloco provisionados pelo Trident suportam os seguintes modos de acesso:

- `ReadWriteOnce` (RWO)
- `ReadOnlyMany` (ROX)
- `ReadWriteOncePod` (RWOP)
- `ReadWriteMany` (RWX), suportado somente quando `volumeMode: Block`

Comportamento do volumeMode

O `volumeMode` campo controla como um volume de bloco é exposto:

- `Filesystem` Trident formata e monta o volume.
- `Block` Trident conecta o dispositivo e o expõe como um dispositivo de bloco raw.

Operações suportadas

Volumes de bloco provisionados usando o driver `google-cloud-netapp-volumes-san` oferecem suporte a:

- Criar
- Excluir
- Clonar
- Snapshot
- Redimensionar
- Importar

Comportamento de sobreprovisionamento de GiB extra

Os volumes em bloco do Google Cloud NetApp Volumes incluem sobrecarga de metadados internos. Essa sobrecarga reduz o tamanho do dispositivo visível para o kernel em comparação com a capacidade provisionada.

Os testes mostram:

- Aproximadamente 300 KiB de sobrecarga na criação inicial
- Até aproximadamente 107 MiB de overhead após um redimensionamento

Como o Google Cloud NetApp Volumes aceita apenas alocações de GiB inteiros, Trident garante que o tamanho utilizável do dispositivo sempre atenda ou exceda a solicitação do PVC por:

- Arredondando o tamanho solicitado para o próximo GiB inteiro
- Adicionando um buffer adicional de 1 GiB

Exemplo:

- Pedido de PVC: 100 GiB
- Tamanho provisionado no Google Cloud NetApp Volumes: 101 GiB

- Espaço utilizável visível para a aplicação: pelo menos 100 GiB

Exemplos de pods

Volume de bloco montado no sistema de arquivos (RWO)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-rwo
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeMounts:
    - name: data
      mountPath: /mnt/data
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-rwo
```

Dispositivo de bloco bruto (RWX)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-raw-rwx
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeDevices:
    - name: data
      devicePath: /dev/xda
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-raw-rwx
```

Comportamento de attach e montagem

Para volumes SAN provisionados a partir do Google Cloud NetApp Volumes:

- Trident cria um Número de Unidade Lógica (LUN) em um pool de storage Flex Unified.
- Durante a publicação, Trident mapeia o LUN para um grupo de hosts por nó.
- Durante o preparo do nó, Trident:
  - Faz login no destino iSCSI
  - Descobre o LUN
  - Configura multipath
- Se `volumeMode: Filesystem`, o Trident formata o dispositivo, se necessário, e o monta.
- Se `volumeMode: Block`Trident anexar o dispositivo e expô-lo diretamente ao pod sem formatar ou montar.

> Os volumes de bloco SAN não oferecem bloqueio distribuído nem coordenação de gravação. Quando um volume de bloco é acessado por vários nós (ReadWriteMany com `volumeMode: Block`), o aplicativo ou o sistema de arquivos deve gerenciar a concorrência.

Prepare-se para configurar um backend do Google Cloud NetApp Volumes

Antes de configurar seu backend do Google Cloud NetApp Volumes, você precisa garantir que os seguintes requisitos sejam atendidos.

Pré-requisitos para volumes NFS ou SMB

Se você estiver usando o Google Cloud NetApp Volumes pela primeira vez ou em um novo local, será necessária alguma configuração inicial para configurar o Google Cloud NetApp Volumes e criar um volume NFS ou SMB. Consulte "Antes de começar".

Certifique-se de ter o seguinte antes de configurar o backend do Google Cloud NetApp Volumes:

- Uma conta do Google Cloud configurada com o serviço Google Cloud NetApp Volumes. Consulte "Google Cloud NetApp Volumes".
- Número do projeto da sua conta do Google Cloud. Consulte "Identificação de projetos".
- Uma conta de serviço do Google Cloud com a função de NetApp Volumes Admin (`roles/netapp.admin`. Consulte "Funções e permissões de Identity and Access Management".
- Arquivo de chave API para sua conta GCNV. Consulte "Criar uma chave de conta de serviço"
- Um pool de storage. Consulte "Visão geral dos storage pools".

Para obter mais informações sobre como configurar o acesso ao Google Cloud NetApp Volumes, consulte "Configure o acesso ao Google Cloud NetApp Volumes".

Opções e exemplos de configuração do backend do Google Cloud NetApp Volumes

Saiba mais sobre as opções de configuração de back-end para Google Cloud NetApp Volumes e veja exemplos de configuração.

Opções de configuração do backend

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

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do driver de armazenamento	O valor de `storageDriverName` deve ser especificado como "google-cloud-netapp-volumes".
`backendName`	(Opcional) Nome personalizado do storage backend	Nome do driver + "_" + parte da chave da API
`storagePools`	Parâmetro opcional usado para especificar pools de storage para criação de volumes.	
`projectNumber`	Número do projeto da conta do Google Cloud. O valor é encontrado na página inicial do portal do Google Cloud.	
`location`	O local do Google Cloud onde Trident cria volumes GCNV. Ao criar clusters Kubernetes entre regiões, volumes criados em um `location` podem ser usados em cargas de trabalho agendadas em nós em várias regiões do Google Cloud. O tráfego entre regiões gera um custo adicional.	
`apiKey`	Chave de API para a conta de serviço do Google Cloud com a `netapp.admin` função. Inclui o conteúdo formatado em JSON do arquivo de chave privada de uma conta de serviço do Google Cloud (copiado integralmente para o arquivo de configuração do backend). O `apiKey` deve incluir pares de chave-valor para as seguintes chaves: `type`, `project_id`, `client_email`, `client_id`, `auth_uri`, `token_uri`, `auth_provider_x509_cert_url` e `client_x509_cert_url`.	
`nfsMountOptions`	Controle preciso das opções de montagem NFS.	"nfsvers=3"
`limitVolumeSize`	Falhe no provisionamento se o tamanho do volume solicitado for superior a esse valor.	"" (não aplicado por padrão)
`serviceLevel`	O nível de serviço de um pool de storage e seus volumes. Os valores são `flex`, `standard`, `premium`, ou `extreme`.	
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar aos volumes	""
`network`	Rede Google Cloud usada para volumes GCNV.	
`debugTraceFlags`	Sinalizadores de depuração para usar na resolução de problemas. Exemplo, `{"api":false, "method":true}`. Não use isso a menos que esteja solucionando problemas e precise de um despejo de log detalhado.	null

Parâmetro	Descrição	Padrão
`nasType`	Configurar a criação de volumes NFS ou SMB. As opções são `nfs`, `smb` ou null. Definir como null define volumes NFS por padrão.	`nfs`
`supportedTopologies`	Representa uma lista de regiões e zonas suportadas por este backend. Para obter mais informações, consulte "Usar a topologia CSI". Por exemplo: `supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a`	

Opções de provisionamento de volume

Você pode controlar o provisionamento de volume padrão na seção `defaults` do arquivo de configuração.

Parâmetro	Descrição	Padrão
`exportRule`	As regras de exportação para novos volumes. Deve ser uma lista separada por vírgulas de qualquer combinação de endereços IPv4.	"0.0.0.0/0"
`snapshotDir`	Acesso ao `.snapshot` diretório	`true`, `false` (o comportamento padrão pode variar. Defina explicitamente) "false" para NFSv3
`snapshotReserve`	Percentual do volume reservado para snapshots	"" (aceitar padrão de 0)
`unixPermissions`	As permissões unix de novos volumes (4 dígitos octais).	""

Exemplos de configurações

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

Configuração mínima

Esta é a configuração mínima absoluta de backend. Com esta configuração, Trident descobre todos os seus pools de storage delegados ao Google Cloud NetApp Volumes no local configurado e coloca novos volumes em um desses pools aleatoriamente. Como `nasType` foi omitido, o `nfs` padrão se aplica e o backend irá provisionar volumes NFS.

Essa configuração é ideal para quem está começando a usar o Google Cloud NetApp Volumes e testando seus recursos, mas na prática pode ser necessário definir um escopo adicional para os volumes provisionados.

Substitua `<id_value>` e `<key_value>` pelas credenciais da sua conta de serviço.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

Configuração para volumes SMB

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    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/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

Configuração com filtro StoragePools

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  storagePools:
    - premium-pool1-europe-west6
    - premium-pool2-europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

Configuração de pool virtual

Esta configuração de backend define vários pools virtuais em um único arquivo. Os pools virtuais são definidos na `storage` seção. Eles são úteis quando você tem vários pools de storage suportando diferentes níveis de serviço e deseja criar classes de storage no Kubernetes que os representem. Os rótulos dos pools virtuais são usados para diferenciá-los. Por exemplo, no exemplo abaixo `performance` label e `serviceLevel` type são usados para diferenciar os pools virtuais.

Você também pode definir alguns valores padrão que serão aplicáveis a todos os pools virtuais e sobrescrever os valores padrão para pools virtuais individuais. No exemplo a seguir, `snapshotReserve` e `exportRule` servem como padrões para todos os pools virtuais.

Para obter mais informações, consulte "Pools virtuais".

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
        performance: standard
      serviceLevel: standard
```

Identidade na nuvem para GKE

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

Configuração de topologias suportadas

Trident facilita o provisionamento de volumes para cargas de trabalho com base em regiões e zonas de disponibilidade. O `supportedTopologies` bloco nesta configuração de backend é usado para fornecer uma lista de regiões e zonas por backend. Os valores de região e zona especificados aqui devem corresponder aos valores de região e zona dos rótulos em cada nó de cluster Kubernetes. Essas regiões e zonas representam a lista de valores permitidos que podem ser fornecidos em uma classe de armazenamento. Para classes de armazenamento que contêm um subconjunto das regiões e zonas fornecidas em um backend, Trident cria volumes na região e zona mencionadas. Para obter mais informações, consulte "Usar a topologia CSI".

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-a
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-b
```

Qual é o próximo passo?

Após criar o arquivo de configuração de backend, execute o seguinte comando:

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

Para verificar se o backend foi criado com sucesso, execute o seguinte comando:

```
kubectl get tridentbackendconfig

NAME                BACKEND NAME        BACKEND UUID
PHASE   STATUS
backend-tbc-gcnv    backend-tbc-gcnv    b2fd1ff9-b234-477e-88fd-713913294f65
Bound   Success
```

Se a criação do backend falhar, há algo errado com a configuração do backend. Você pode descrever o backend usando o `kubectl get tridentbackendconfig <backend-name>` comando ou visualizar os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs
```

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

Definições de classe de armazenamento

A seguir está uma definição básica `StorageClass` que se refere ao backend acima.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

Exemplo de definições usando o `parameter.selector` campo:

Usando `parameter.selector` você pode especificar para cada `StorageClass` o "pool virtual" que é usado para hospedar um volume. O volume terá os aspectos definidos no pool escolhido.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=standard
  backendType: google-cloud-netapp-volumes
```

Para mais detalhes sobre classes de armazenamento, consulte "Crie uma storage class".

Exemplos de definições para volumes SMB

Usando `nasType`, `node-stage-secret-name` e `node-stage-secret-namespace`, você pode especificar um volume SMB e fornecer as credenciais necessárias do Active Directory. Qualquer usuário/senha do Active Directory com qualquer/nenhuma permissão pode ser usado para o segredo do estágio do nó.

Configuração básica no namespace padrão

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

Usando segredos diferentes por namespace

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

Utilizando segredos diferentes em cada volume

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

`nasType: smb`Filtros para pools que suportam volumes SMB. `nasType: nfs` ou `nasType: null`filtros para pools NFS.

Exemplo de definição de PVC

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc
```

Para verificar se o PVC está vinculado, execute o seguinte comando:

```
kubectl get pvc gcnv-nfs-pvc

NAME          STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS AGE
gcnv-nfs-pvc  Bound    pvc-b00f2414-e229-40e6-9b16-ee03eb79a213  100Gi
RWX            gcnv-nfs-sc  1m
```

Configurar o auto-tiering para Google Cloud NetApp Volumes

O auto-tiering é configurado por meio dos parâmetros do backend do Trident e das anotações PersistentVolumeClaim durante o provisionamento de volumes. Você pode configurar o auto-tiering para Google Cloud NetApp Volumes usando Trident.

Visão geral

O recurso de tiering automático permite que o Trident provisione volumes que movem automaticamente dados inativos de uma camada de desempenho para uma camada de capacidade. Isso reduz o custo de armazenamento enquanto preserva o desempenho para dados acessados com frequência.

Trident aplica as configurações de armazenamento em camadas automático somente no momento da criação do volume. Alterações posteriores ao provisionamento não são suportadas em Trident 26.02.

Conceitos

Tiering automático

O auto-tiering move dados acessados com pouca frequência de uma camada de desempenho para uma camada de capacidade com base em padrões de acesso. A movimentação de dados ocorre de forma

assíncrona e não é imediata.

Política de tiering

A política de tiering determina se o auto-tiering está habilitado para um volume.

As seguintes políticas são suportadas: * `auto`: ativa o tiering automático com base em padrões de acesso * `none`: desativa o tiering automático

Dias de resfriamento

Os dias de resfriamento especificam o número mínimo de dias que um bloco de dados deve permanecer inativo antes de se tornar elegível para o armazenamento em camadas. Os dias de resfriamento se aplicam somente quando a política de armazenamento em camadas está definida como `auto`.

Modelo de configuração

Escopos de configuração

O auto-tiering pode ser configurado em vários escopos:

- **Escopo do pool de armazenamento** Aplica-se a todos os volumes provisionados do pool.
- **Escopo do volume** Aplica-se a um único volume por meio de anotações PersistentVolumeClaim.

Trident determina a configuração efetiva com base em onde cada configuração está definida.

Precedência de configuração

Quando a mesma configuração é definida em vários escopos, Trident aplica a seguinte ordem de precedência:

1. Anotações de PersistentVolumeClaim
2. Configuração do backend Trident
3. Padrões do pool de armazenamento

As configurações definidas em uma precedência mais alta substituem os valores de nível inferior.

Funcionalidade suportada no Trident 26.02

Trident 26.02 oferece suporte aos seguintes recursos de auto-tiering para Google Cloud NetApp Volumes:

- Habilitar ou desabilitar o auto-tiering durante o provisionamento de volumes
- Definindo uma política de hierarquização na configuração do backend do Trident
- Substituindo a política de escalonamento e os dias de resfriamento por volume usando anotações de PVC
- Configurando dias de resfriamento para volumes com auto-tiering ativado

Funcionalidade não suportada no Trident 26.02

As seguintes operações não são suportadas:

- Modificando as configurações de auto-tiering após a criação do volume
- Alterando políticas de camadas em volumes existentes usando atualizações do Kubernetes

- Aplicando configurações de armazenamento em camadas automático fora dos fluxos de trabalho de provisionamento gerenciados pelo Trident

Parâmetros de configuração do backend

Os seguintes parâmetros controlam o comportamento de auto-tiering quando definidos na configuração do backend Trident:

Parâmetro	Obrigatório	Descrição
tieringPolicy	Não	Política de escalonamento para volumes (`auto` ou `none`)
tieringMinimumCoolingDays	Não	Número de dias de inatividade antes dos dados serem transferidos de camada (intervalo: 2–183, padrão: 31)

Substituições em nível de volume usando PersistentVolumeClaim anotações

Anotações suportadas

PersistentVolumeClaim anotações permitem a substituição das configurações de auto-tiering por volume.

Anotação	Descrição
`trident.netapp.io/tieringPolicy`	Substitui a política de hierarquização para o volume
`trident.netapp.io/tieringMinimumCoolingDays`	Substitui o valor dos dias de resfriamento para o volume

Exemplo: PersistentVolumeClaim com substituições de hierarquização automática

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-tiering-pvc
  annotations:
    trident.netapp.io/tieringPolicy: auto
    trident.netapp.io/tieringMinimumCoolingDays: "45"
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: google-cloud-netapp-volumes-auto-tiering
  resources:
    requests:
      storage: 500Gi
```

Comportamento e limitações

Comportamento de provisionamento

- As configurações de auto-tiering são avaliadas e aplicadas somente no momento da criação do volume.
- Trident não reconcilia a configuração de tiering após o provisionamento.
- Os dias de resfriamento são ignorados quando a política de tiering está definida como `none`.

Limitações da plataforma

- O auto-tiering é compatível apenas com volumes NAS (NFS e SMB).
- Volumes em bloco (iSCSI) não suportam auto-tiering.
- O pool de storage do Google Cloud NetApp Volumes deve ter o armazenamento em camadas automático ativado no Google Cloud.

Valores suportados

- Intervalo válido para `tieringMinimumCoolingDays`: 2 a 183
- Valor padrão: 31

Configurar um NetApp HCI ou SolidFire backend

Aprenda como criar e usar um backend Element com sua instalação do Trident.

Detalhes do driver Element

Trident fornece o `solidfire-san` storage driver para se comunicar com o cluster. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

O `solidfire-san` driver de armazenamento suporta os modos de volume *file* e *block*. Para o `Filesystem` volumeMode, Trident cria um volume e cria um sistema de arquivos. O tipo de sistema de arquivos é especificado pelo StorageClass.

Driver	Protocolo	VolumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`solidfire-san`	iSCSI	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de arquivos. Dispositivo de bloco bruto.
`solidfire-san`	iSCSI	Sistema de arquivos	RWO, RWOP	`xfs`, `ext3`, `ext4`

Antes de começar

Você precisará do seguinte antes de criar um backend Element.

- Um sistema de storage compatível que execute o software Element.
- Credenciais para um administrador de cluster NetApp HCI/SolidFire ou usuário de locatário que possa gerenciar volumes.

- Todos os seus nós de trabalho do Kubernetes devem ter as ferramentas iSCSI apropriadas instaladas. Consulte "Informações sobre preparação do nó de trabalho".

Opções de configuração do backend

Consulte a tabela a seguir para as opções de configuração do backend:

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do driver de armazenamento	Sempre "solidfire-san"
`backendName`	Nome personalizado ou o storage backend	"solidfire_" + endereço IP de storage (iSCSI)
`Endpoint`	MVIP para o SolidFire cluster com credenciais de locatário	
`SVIP`	Endereço IP e porta de storage (iSCSI)	
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar aos volumes.	""
`TenantName`	Nome do locatário a ser usado (criado se não for encontrado)	
`InitiatorIFace`	Restringir o tráfego iSCSI a uma interface de host específica	"default"
`UseCHAP`	Use CHAP para autenticar iSCSI. Trident usa CHAP.	verdadeiro
`AccessGroups`	Lista de IDs de grupos de acesso a serem usados	Encontra o ID de um grupo de acesso chamado "trident"
`Types`	Especificações de QoS	
`limitVolumeSize`	Falhar no provisionamento se o tamanho do volume solicitado for superior a este valor	"" (não aplicado por padrão)
`debugTraceFlags`	Sinalizadores de depuração para usar na resolução de problemas. Exemplo, {"api":false, "method":true}	null

AVISO Não utilize `debugTraceFlags` a menos que esteja solucionando problemas e necessite de um despejo de logs detalhado.

Exemplo 1: configuração de backend para `solidfire-san` driver com três tipos de volume

Este exemplo mostra um arquivo de backend usando autenticação CHAP e modelando três tipos de volume com garantias de QoS específicas. Muito provavelmente, você definiria classes de armazenamento para consumir cada um deles 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 backend e classe de armazenamento para `solidfire-san`driver com pools virtuais

Este exemplo mostra o arquivo de definição de backend configurado com pools virtuais juntamente com StorageClasses que fazem referência a eles.

Trident copia os rótulos presentes em um pool de storage para o LUN de storage de backend durante o provisionamento. Para conveniência, administradores de storage podem definir rótulos por pool virtual e agrupar volumes por rótulo.

No arquivo de definição de backend de exemplo mostrado abaixo, valores padrão específicos são definidos para todos os pools de storage, que definem o `type` em Silver. Os pools virtuais são definidos na seção `storage`. Neste exemplo, alguns pools de storage definem seu próprio tipo e alguns pools 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 de StorageClass referem-se aos pools virtuais acima. Usando o campo `parameters.selector`, cada StorageClass indica qual(is) pool(s) virtual(is) pode(m) ser usado(s) para

hospedar um volume. O volume terá os aspectos definidos no pool virtual escolhido.

O primeiro StorageClass (`solidfire-gold-four` mapeará o primeiro pool virtual. Este é o único pool que oferece desempenho Gold com um `Volume Type QoS` de Gold. O último StorageClass (`solidfire-silver` indica qualquer pool de storage que ofereça desempenho Silver. Trident decidirá qual pool virtual será 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 a volume"

Drivers SAN do ONTAP

Visão geral do driver ONTAP SAN

Saiba mais sobre como configurar um backend ONTAP com os drivers SAN do ONTAP e do Cloud Volumes ONTAP.

Detalhes do driver ONTAP SAN

Trident fornece os seguintes drivers de armazenamento SAN para se comunicar com o ONTAP cluster. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Driver	Protocolo	volumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`ontap-san`	iSCSI SCSI sobre FC	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de arquivos; dispositivo de bloco bruto
`ontap-san`	iSCSI SCSI sobre FC	Sistema de arquivos	RWO, RWOP ROX e RWX não estão disponíveis no modo de volume de sistema de arquivos.	`xfs`, `ext3`, `ext4`
`ontap-san`	NVMe/TCP Consulte Considerações adicionais para NVMe/TCP.	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de arquivos; dispositivo de bloco bruto

Driver	Protocolo	volumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`ontap-san`	NVMe/TCP Consulte Considerações adicionais para NVMe/TCP.	Sistema de arquivos	RWO, RWOP ROX e RWX não estão disponíveis no modo de volume de sistema de arquivos.	`xfs`, `ext3`, `ext4`
`ontap-san-economy`	iSCSI	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de arquivos; dispositivo de bloco bruto
`ontap-san-economy`	iSCSI	Sistema de arquivos	RWO, RWOP ROX e RWX não estão disponíveis no modo de volume de sistema de arquivos.	`xfs`, `ext3`, `ext4`

AVISO

- Use `ontap-san-economy` somente se a contagem de uso de volume persistente prevista for maior que "limites de volume ONTAP suportados".
- Use `ontap-nas-economy` somente se a contagem de uso de volume persistente prevista for maior que "limites de volume ONTAP suportados" e o driver `ontap-san-economy` não puder ser usado.
- Não utilize `ontap-nas-economy` se você prevê a necessidade de proteção de dados, recuperação de desastres ou mobilidade.
- NetApp não recomenda usar o crescimento automático do FlexVol em todos os drivers ONTAP, exceto ontap-san. Como alternativa, Trident oferece suporte ao uso de reserva de snapshot e dimensiona os volumes FlexVol de acordo.

Permissões do usuário

Trident espera ser executado como administrador do ONTAP ou do SVM, normalmente usando o `admin` usuário do cluster ou um `vsadmin` usuário do SVM, ou um usuário com um nome diferente que tenha a mesma função. Para implantações do Amazon FSx for NetApp ONTAP, Trident espera ser executado como administrador do ONTAP ou do SVM, usando o usuário do cluster `fsxadmin` ou um usuário do SVM `vsadmin`, 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.

OBSERVAÇÃO

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

Embora seja possível criar uma função mais restritiva dentro do ONTAP que um driver Trident possa usar, não recomendamos isso. A maioria das novas versões do Trident chamará APIs adicionais que precisariam ser

consideradas, dificultando as atualizações e tornando-as propensas a erros.

Considerações adicionais para NVMe/TCP

Trident suporta o protocolo non-volatile memory express (NVMe) usando o `ontap-san` driver incluindo:

- IPv6
- Instantâneos e clones de volumes NVMe
- Redimensionando um volume NVMe
- Importando um volume NVMe criado fora do Trident para que seu ciclo de vida possa ser gerenciado pelo Trident
- Multipathing nativo NVMe
- Encerramento correto ou incorreto dos nós K8s (24.06)

Trident não suporta:

- DH-HMAC-CHAP que é suportado nativamente pelo NVMe
- Multipathing do device mapper (DM)
- LUKS criptografia

OBSERVAÇÃO	O NVMe é suportado apenas com as APIs REST do ONTAP e não é suportado com ONTAPI (ZAPI).

Prepare-se para configurar o backend com os drivers ONTAP SAN

Compreenda os requisitos e as opções de autenticação para configurar um backend ONTAP com drivers ONTAP SAN.

Requisitos

Para todos os backends ONTAP, Trident exige que pelo menos um agregado seja atribuído à SVM.

OBSERVAÇÃO	"Sistemas ASA r2" diferem de outros sistemas ONTAP (ASA, AFF e FAS) na implementação de sua camada de storage. Nos sistemas ASA r2, zonas de disponibilidade de storage são usadas em vez de agregados. Consulte o "este" artigo da Knowledge Base sobre como atribuir agregados a SVMs em sistemas ASA r2.

Lembre-se de que você também pode executar mais de um driver e criar classes de armazenamento que apontem 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 o `ontap-san-economy` driver.

Todos os seus nós de trabalho do Kubernetes devem ter as ferramentas iSCSI apropriadas instaladas. Consulte "Prepare o nó de trabalho" para obter detalhes.

Autenticar o backend ONTAP

Trident oferece dois modos de autenticação de um backend ONTAP.

- Baseado em credenciais: o nome de usuário e a senha de um usuário do ONTAP com as permissões

necessárias. Recomenda-se o uso de uma função de login de segurança predefinida, como `admin` ou `vsadmin` para garantir a máxima compatibilidade com as versões do ONTAP.

- Com base em certificado: Trident também pode se comunicar com um ONTAP cluster usando um certificado instalado no backend. Nesse caso, a definição do backend deve conter os valores codificados em Base64 do certificado do cliente, da chave e do certificado da CA confiável, se utilizado (recomendado).

Você pode atualizar os backends existentes para alternar entre métodos baseados em credenciais e baseados em certificados. No entanto, apenas um método de autenticação é suportado por vez. Para mudar para um método de autenticação diferente, você deve remover o método existente da configuração do backend.

AVISO	Se você tentar fornecer **tanto credenciais quanto certificados**, a criação do backend falhará com um erro informando que mais de um método de autenticação foi fornecido no arquivo de configuração.

Ativar autenticação baseada em credenciais

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

Uma definição de backend de exemplo será semelhante a esta:

YAML

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

JSON

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

Lembre-se de que a definição do backend é o único local onde as credenciais são armazenadas em texto simples. Após a criação do backend, nomes de usuário/senhas são codificados em Base64 e armazenados como segredos do Kubernetes. A criação ou atualização de um backend é a única etapa que requer conhecimento das credenciais. Assim, trata-se de uma operação exclusiva do administrador, a ser realizada pelo administrador de storage do Kubernetes.

Ativar autenticação baseada em certificado

Novos e existentes backends podem usar um certificado e se comunicar com o backend do ONTAP. Três parâmetros são necessários na definição do backend.

- clientCertificate: Valor codificado em Base64 do certificado do cliente.
- clientPrivateKey: Valor codificado em Base64 da chave privada associada.
- trustedCACertificate: valor codificado em Base64 do certificado CA confiável. Se uma CA confiável estiver sendo usada, este parâmetro deve ser fornecido. Isso pode ser ignorado se nenhuma CA confiável for usada.

Um fluxo de trabalho típico envolve as seguintes etapas.

Passos

1. Gere um certificado de cliente e uma chave. Ao gerar, defina o Nome Comum (CN) para o usuário ONTAP que será usado para autenticação.

```
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. Adicione um certificado de CA confiável ao cluster ONTAP. Isso pode já ter sido configurado pelo administrador de storage. Ignore se nenhuma CA confiável for utilizada.

```
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 do cliente e a chave (do passo 1) no cluster ONTAP.

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

OBSERVAÇÃO	Após executar este comando, ONTAP solicitará a entrada do certificado. Cole o conteúdo do `k8senv.pem` arquivo gerado na etapa 1, depois pressione `END` para concluir a instalação.

4. Confirme se a função de login de segurança do ONTAP suporta `cert` 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 o certificado gerado. Substitua <ONTAP Management LIF> e <vserver name> pelo endereço IP da Management LIF e pelo 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. Codifique o certificado, a chave e o certificado da 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 o backend usando os valores obtidos na etapa anterior.

```
cat cert-backend.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Atualize os métodos de autenticação ou altere as credenciais

Você pode atualizar um backend existente para usar um método de autenticação diferente ou para rotacionar suas credenciais. Isso funciona nos dois sentidos: backends que utilizam nome de usuário/senha podem ser atualizados para usar certificados; backends que utilizam certificados podem ser atualizados para usar nome de usuário/senha. Para fazer isso, você deve remover o método de autenticação existente e adicionar o novo método de autenticação. Em seguida, use o arquivo backend.json atualizado contendo os parâmetros necessários para executar `tridentctl backend update`.

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

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

OBSERVAÇÃO	Ao rotacionar senhas, o administrador de storage deve primeiro atualizar a senha do usuário no ONTAP. Em seguida, é feita uma atualização no backend. Ao rotacionar certificados, vários certificados podem ser adicionados ao usuário. O backend é então atualizado para usar o novo certificado, após o que o certificado antigo pode ser excluído do cluster ONTAP.

A atualização do backend não interrompe o acesso aos volumes já criados, nem afeta as conexões de volume feitas posteriormente. Uma atualização bem-sucedida do backend indica que Trident pode se comunicar com o ONTAP backend e lidar com operações de volume futuras.

Criar função ONTAP personalizada para Trident

Você pode criar uma função de cluster ONTAP com privilégios mínimos para que não precise usar a função de administrador do ONTAP para executar operações no Trident. Ao incluir o nome de usuário em um arquivo de configuração de backend do Trident, o Trident usa a função de cluster ONTAP que você criou para executar as operações.

Consulte "Gerador de funções personalizadas Trident" para obter mais informações sobre como criar funções personalizadas do Trident.

Usando ONTAP CLI

1. Crie uma nova função usando o seguinte comando:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Crie um nome de usuário para o usuário do Trident:

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Mapeie a função para o usuário:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Usando System Manager

Execute as seguintes etapas no ONTAP System Manager:

1. **Crie uma função personalizada**:
   a. Para criar uma função personalizada no nível do cluster, selecione **Cluster > Settings**.

      (Ou) Para criar uma função personalizada no nível da SVM, selecione **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**.

   b. Selecione o ícone de seta (→) ao lado de **Users and Roles**.
   c. Selecione **+Adicionar** em **Roles**.
   d. Defina as regras para a função e clique em **Save**.
2. **Mapeie a função ao usuário Trident**: + Execute as seguintes etapas na página **Usuários e Funções**:
   a. Selecione o ícone Adicionar **+** em **Usuários**.
   b. Selecione o nome de usuário desejado e selecione uma função no menu suspenso para **Função**.
   c. Clique em **Salvar**.

Consulte as seguintes páginas para obter mais informações:

- "Funções personalizadas para administração do ONTAP" ou "Definir funções personalizadas"
- "Trabalhe com funções e usuários"

Autentique conexões com CHAP bidirecional

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

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

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

Além de `useCHAP=true`, os campos `chapInitiatorSecret`, `chapTargetInitiatorSecret`, `chapTargetUsername` e `chapUsername` devem ser incluídos na definição do backend. Os segredos podem ser alterados após a criação de um backend executando `tridentctl update`.

Como funciona

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

- Configurando CHAP no SVM:
  - Se o tipo de segurança do iniciador padrão da SVM for none (definido por padrão) **e** não houver LUNs preexistentes no volume, Trident definirá o tipo de segurança padrão para `CHAP` e prosseguirá com a configuração do nome de usuário e segredos do iniciador e destino CHAP.
  - Se a SVM contiver LUNs, Trident não habilitará CHAP na SVM. Isso garante que o acesso às LUNs já presentes na SVM não seja restringido.
- Configuração do nome de usuário e dos segredos do iniciador e do alvo CHAP; essas opções devem ser especificadas na configuração do backend (como mostrado acima).

Após a criação do backend, Trident cria um correspondente `tridentbackend` CRD e armazena os segredos CHAP e os nomes de usuário como segredos do Kubernetes. Todos os PVs que são criados pelo Trident nesse backend serão montados e anexados via CHAP.

Rotacionar credenciais e atualizar backends

Você pode atualizar as credenciais CHAP atualizando os parâmetros CHAP no `backend.json` arquivo. Isso exigirá atualizar os segredos CHAP e usar o `tridentctl update` comando para refletir essas alterações.

AVISO Ao atualizar os segredos CHAP de um backend, você deve usar `tridentctl` para atualizar o backend. Não atualize as credenciais no cluster de storage usando a ONTAP CLI ou ONTAP System Manager, pois Trident não conseguirá detectar 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": "password",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

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

As conexões existentes permanecerão inalteradas; elas continuarão ativas se as credenciais forem atualizadas pelo Trident no SVM. Novas conexões usam as credenciais atualizadas e as conexões existentes continuam ativas. Desconectar e reconectar PVs antigos fará com que eles usem as credenciais atualizadas.

Opções e exemplos de configuração do ONTAP SAN

Aprenda como criar e usar drivers ONTAP SAN com sua instalação do Trident. Esta seção fornece exemplos de configuração de backend e detalhes para mapear backends para StorageClasses. "Sistemas ASA r2" diferem de outros sistemas ONTAP (ASA, AFF e FAS) na implementação de sua camada de storage. Essas variações impactam o uso de certos parâmetros conforme indicado. "Saiba mais sobre as diferenças entre sistemas ASA r2 e outros sistemas ONTAP". Na configuração do backend do Trident, não é necessário especificar que seu sistema é ASA r2. Ao selecionar `ontap-san` como o

`storageDriverName`, o Trident detecta automaticamente os sistemas ASA r2 ou outros sistemas ONTAP. Alguns parâmetros de configuração do backend não se aplicam a sistemas ASA r2, conforme indicado na tabela abaixo.

OBSERVAÇÃO	Apenas o `ontap-san`driver (com os protocolos iSCSI, NVMe/TCP e FC) é compatível com sistemas ASA r2.

Opções de configuração do backend

Consulte a tabela a seguir para as opções de configuração do backend:

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do driver de armazenamento	`ontap-san` ou `ontap-san-economy`
`backendName`	Nome personalizado ou o storage backend	Nome do driver + "_" + dataLIF
`managementLIF`	Endereço IP de um cluster ou LIF de gerenciamento de SVM. É possível especificar um nome de domínio totalmente qualificado (FQDN). Pode ser configurado para usar endereços IPv6 se Trident foi instalado com o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Para um switchover MetroCluster perfeito, consulte o Exemplo do MetroCluster. **OBSERVAÇÃO** Se estiver usando credenciais "vsadmin", `managementLIF` deve ser a da SVM; se estiver usando credenciais "admin", `managementLIF` deve ser a do cluster.	"10.0.0.1", "[2001:1234:abcd::fefe]"

Parâmetro	Descrição	Padrão
`dataLIF`	Endereço IP do protocolo LIF. Pode ser configurado para usar endereços IPv6 se Trident foi instalado com o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **Não especifique para iSCSI.** Trident usa "Mapa de LUN seletivo do ONTAP" para descobrir os LIFs iSCSI necessários para estabelecer uma sessão de múltiplos caminhos. Um aviso é gerado se `dataLIF` for definido explicitamente. **Omita para MetroCluster.** Consulte o Exemplo do MetroCluster.	Derivado pelo SVM
`svm`	Máquina virtual de storage a ser usada **Omitir para MetroCluster.** Consulte o Exemplo do MetroCluster.	Derivado se uma SVM `managementLIF` for especificada
`useCHAP`	Use CHAP para autenticar iSCSI para drivers ONTAP SAN [parâmetro booleano]. Defina como `true` para que Trident configure e use CHAP bidirecional como autenticação padrão para o SVM fornecido no backend. Consulte "Prepare-se para configurar o backend com os drivers ONTAP SAN" para obter detalhes. **Não compatível com FCP ou NVMe/TCP.**	`false`
`chapInitiatorSecret`	Segredo do iniciador CHAP. Obrigatório se `useCHAP=true`	""
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar aos volumes	""
`chapTargetInitiatorSecret`	Segredo do iniciador do alvo CHAP. Obrigatório se `useCHAP=true`	""
`chapUsername`	Nome de usuário de entrada. Obrigatório se `useCHAP=true`	""
`chapTargetUsername`	Nome de usuário de destino. Obrigatório 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 da CA confiável. Opcional. Usado para autenticação baseada em certificado.	""
`username`	Nome de usuário necessário para se comunicar com o cluster ONTAP. Usada para autenticação baseada em credenciais. Para autenticação do Active Directory, consulte "Autentique o Trident em um SVM de backend usando credenciais do Active Directory".	""

Parâmetro	Descrição	Padrão
`password`	Senha necessária para se comunicar com o cluster ONTAP. Usada para autenticação baseada em credenciais. Para autenticação do Active Directory, consulte "Autentique o Trident em um SVM de backend usando credenciais do Active Directory".	""
`svm`	Máquina virtual de storage para usar	Derivado se uma SVM `managementLIF` for especificada
`storagePrefix`	Prefixo usado ao provisionar novos volumes no SVM. Não pode ser modificado posteriormente. Para atualizar este parâmetro, você precisará criar um novo backend.	`trident`
`aggregate`	Agregado para provisionamento (opcional; se definido, deve ser atribuído à SVM). Para o `ontap-nas-flexgroup` driver, esta opção é ignorada. Se não for atribuído, qualquer um dos agregados disponíveis pode ser usado para provisionar um FlexGroup volume. **OBSERVAÇÃO** Quando o agregado é atualizado no SVM, ele é atualizado automaticamente no Trident por meio de polling no SVM, sem a necessidade de reiniciar o Trident Controller. Quando você configurou um agregado específico no Trident para provisionar volumes, se o agregado for renomeado ou movido para fora do SVM, o backend entrará em estado de falha no Trident durante o polling do agregado no SVM. Você deve alterar o agregado para um que esteja presente no SVM ou removê-lo completamente para que o backend volte a ficar online. **Não especificar para sistemas ASA r2**.	""

Parâmetro	Descrição	Padrão
`limitAggregateUsage`	O provisionamento falhará se o uso ultrapassar essa porcentagem. Se você estiver usando um Amazon FSx para NetApp ONTAP backend, não especifique `limitAggregateUsage`. As configurações fornecidas `fsxadmin` e `vsadmin` não contêm as permissões necessárias para recuperar o uso agregado e limitá-lo usando Trident. **Não especificar para sistemas ASA r2**.	"" (não aplicado por padrão)
`limitVolumeSize`	O provisionamento falha se o tamanho do volume solicitado for superior a este valor. Também restringe o tamanho máximo dos volumes que gerencia para LUNs.	"" (não aplicado por padrão)
`lunsPerFlexvol`	Número máximo de LUNs por FlexVol, deve estar no intervalo [50, 200]	`100`
`debugTraceFlags`	Sinalizadores de depuração para usar na resolução de problemas. Exemplo, {"api":false, "method":true} Não use a menos que esteja solucionando problemas e precise de um despejo de log detalhado.	`null`

Parâmetro	Descrição	Padrão
`useREST`	Parâmetro booleano para usar as ONTAP REST APIs. `useREST` Quando definido como `true`, Trident usa as ONTAP REST APIs para se comunicar com o backend; quando definido como `false`, Trident usa chamadas ONTAPI (ZAPI) para se comunicar com o backend. Este recurso requer ONTAP 9.11.1 e versões posteriores. Além disso, a função de login do ONTAP utilizada deve ter acesso ao aplicativo `ontapi`. Isso é atendido pelas funções predefinidas `vsadmin` e `cluster-admin`. A partir da versão 24.06 do Trident e ONTAP 9.15.1 ou posterior, `useREST` é definido como `true` por padrão; altere `useREST` para `false` para usar chamadas ONTAPI (ZAPI). `useREST` está totalmente qualificado para NVMe/TCP. **OBSERVAÇÃO** O NVMe é suportado apenas com as APIs REST do ONTAP e não é suportado com ONTAPI (ZAPI). **Se especificado, defina sempre como `true` para sistemas ASA r2**.	`true`para ONTAP 9.15.1 ou posterior, caso contrário `false`.
`sanType`	Use para selecionar `iscsi` para iSCSI, `nvme` para NVMe/TCP ou `fcp` para SCSI sobre Fibre Channel (FC).	`iscsi` se estiver em branco

Parâmetro	Descrição	Padrão
`formatOptions`	Use `formatOptions` para especificar argumentos de linha de comando para o comando `mkfs`, que serão aplicados sempre que um volume for formatado. Isso permite formatar o volume de acordo com suas preferências. Certifique-se de especificar as formatOptions de forma semelhante às opções do comando mkfs, excluindo o caminho do dispositivo. Exemplo: "-E nodiscard" **Compatível para `ontap-san` e `ontap-san-economy`drivers com o protocolo iSCSI. Além disso, compatível com sistemas ASA r2 ao usar os protocolos iSCSI e NVMe/TCP.**	
`limitVolumePoolSize`	Tamanho máximo solicitável de FlexVol ao usar LUNs no backend ontap-san-economy.	"" (não aplicado por padrão)
`denyNewVolumePools`	Restringe `ontap-san-economy`os backends de criar novos volumes FlexVol para conter seus LUNs. Somente FlexVols preexistentes são usados para provisionar novos PVs.	

Recomendações para uso de formatOptions

Trident recomenda as seguintes opções para agilizar o processo de formatação:

- **-E nodiscard (ext3, ext4):** Não tente descartar blocos durante a criação do sistema de arquivos (o descarte inicial de blocos é útil em dispositivos de estado sólido e em storage esparso/com thin provisioning). Esta opção substitui a opção obsoleta "-K" e é aplicável aos sistemas de arquivos ext3 e ext4.
- **-K (xfs):** Não tente descartar blocos durante a criação do sistema de arquivos (mkfs). Esta opção se aplica ao sistema de arquivos xfs.

Autentique o Trident em um SVM de backend usando credenciais do Active Directory

Você pode configurar Trident para autenticar em uma SVM de backend usando credenciais do Active Directory (AD). Antes que uma conta do AD possa acessar a SVM, você deve configurar o acesso do controlador de domínio do AD ao cluster ou à SVM. Para administração do cluster com uma conta do AD, você deve criar domain tunnel. Consulte "Configurar o acesso do controlador de domínio do Active Directory no ONTAP" para obter detalhes.

passos

1. Configurar as definições do Domain Name System (DNS) para uma SVM de backend:

```
vserver services dns create -vserver <svm_name> -dns-servers
<dns_server_ip1>,<dns_server_ip2>
```

2. Execute o seguinte comando para criar uma conta de computador para a SVM no Active Directory:

```
vserver active-directory create -vserver DataSVM -account-name ADSERVER1
-domain demo.netapp.com
```

3. Use este comando para criar um usuário ou grupo do AD para gerenciar o cluster ou SVM

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. No arquivo de configuração do Trident backend, defina os parâmetros `username` e `password` para o nome de usuário ou grupo do AD e a senha, respectivamente.

Opções de configuração de backend para provisionamento de volumes

Você pode controlar o provisionamento padrão usando essas opções na seção `defaults` do arquivo de configuração. Para um exemplo, veja os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
`spaceAllocation`	Alocação de espaço para LUNs	"verdadeiro" **Se especificado, defina como `true` para sistemas ASA r2**.
`spaceReserve`	Modo de reserva de espaço; "nenhum" (com thin provisioning) ou "volume" (thick). **Definido como `none` para sistemas ASA r2**.	"none"
`snapshotPolicy`	Política do Snapshot a ser usada. **Definida como `none` para sistemas ASA r2**.	"none"
`qosPolicy`	Grupo de políticas de QoS a ser atribuído aos volumes criados. Escolha um dos qosPolicy ou adaptiveQosPolicy por pool de storage/backend. O uso de grupos de políticas de QoS com Trident requer ONTAP 9.8 ou posterior. Você deve usar um grupo de políticas de QoS não compartilhado e garantir que o grupo de políticas seja aplicado a cada componente individualmente. Um grupo de políticas de QoS compartilhado impõe o limite máximo para a taxa de transferência total de todas as cargas de trabalho.	""
`adaptiveQosPolicy`	Grupo de políticas de QoS adaptável para atribuir aos volumes criados. Escolha uma de qosPolicy ou adaptiveQosPolicy por pool de storage/backend	""
`snapshotReserve`	Percentual do volume reservado para snapshots. **Não especificar para sistemas ASA r2**.	"0" se `snapshotPolicy` for "none", caso contrário ""
`splitOnClone`	Separar um clone de seu progenitor no momento da criação	"false"
`encryption`	Habilite NetApp Volume Encryption (NVE) no novo volume; o padrão é `false`. A NVE deve estar licenciada e habilitada no cluster para usar esta opção. Se a NAE estiver habilitada no backend, qualquer volume provisionado no Trident terá a NAE habilitada. Para mais informações, consulte: "Como Trident funciona com NVE e NAE".	"falso" **Se especificado, defina como `true` para sistemas ASA r2**.

Parâmetro	Descrição	Padrão
`luksEncryption`	Ative a criptografia LUKS. Consulte "Use Linux Unified Key Setup (LUKS)".	**Defina como `false` para sistemas ASA r2.**
`tieringPolicy`	Política de tiering para usar "none" **Não especificar para sistemas ASA r2.**	
`nameTemplate`	Modelo para criar nomes de volume personalizados.	""

Exemplos de provisionamento de volume

Aqui está um exemplo com valores padrão definidos:

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

OBSERVAÇÃO

Para todos os volumes criados usando o `ontap-san` driver, Trident adiciona 10 por cento de capacidade extra ao FlexVol para acomodar os metadados do LUN. O LUN será provisionado com o tamanho exato que o usuário solicitar no PVC. Trident adiciona 10 por cento ao FlexVol (exibido como tamanho disponível no ONTAP). Os usuários agora receberão a quantidade de capacidade utilizável que solicitaram. Essa alteração também impede que os LUNs se tornem somente leitura, a menos que o espaço disponível esteja totalmente utilizado. Isso não se aplica ao ontap-san-economy.

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

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

O valor 1.1 representa os 10% adicionais que o Trident adiciona ao FlexVol para acomodar os metadados da LUN. Para `snapshotReserve` = 5%, e solicitação de PVC = 5 GiB, o tamanho total do volume é 5,79 GiB e o tamanho disponível é 5,5 GiB. O comando `volume show` deve exibir resultados semelhantes a este exemplo:

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

Exemplos de configuração mínima

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

OBSERVAÇÃO	Se você estiver usando Amazon FSx no NetApp ONTAP com Trident, NetApp recomenda que você especifique nomes DNS para LIFs em vez de endereços IP.

Exemplo de ONTAP SAN

Esta é uma configuração básica usando o `ontap-san` driver.

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

Exemplo do MetroCluster

Você pode configurar o backend para evitar ter que atualizar manualmente a definição do backend após switchover e switchback durante "Replicação e recuperação de SVM".

Para switchover e switchback sem interrupções, especifique a SVM usando `managementLIF` e omita os parâmetros `svm`. Por exemplo:

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

Exemplo de economia ONTAP SAN

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

Exemplo de autenticação baseada em certificado

Neste exemplo de configuração básica `clientCertificate`, `clientPrivateKey` e `trustedCACertificate` (opcional, se estiver usando uma CA confiável) são preenchidos em `backend.json` e recebem os valores codificados em base64 do certificado do cliente, da chave privada e do certificado da CA confiável, respectivamente.

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

Exemplos bidirecionais CHAP

Esses exemplos criam um backend com `useCHAP` definido como `true`.

Exemplo ONTAP SAN CHAP

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

Exemplo de economia SAN CHAP do ONTAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

Exemplo de NVMe/TCP

Você precisa ter uma SVM configurada com NVMe no seu backend ONTAP. Esta é uma configuração básica de backend para NVMe/TCP.

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

Exemplo de SCSI sobre FC (FCP)

Você precisa ter uma SVM configurada com FC no seu ONTAP backend. Esta é uma configuração básica de backend para FC.

```
---
version: 1
backendName: fcp-backend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_fc
username: vsadmin
password: password
sanType: fcp
useREST: true
```

Exemplo de configuração de backend com nameTemplate

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

formatOptions exemplo para o driver ontap-san-economy

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: ""
svm: svm1
username: ""
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: -E nodiscard
```

Exemplos de backends com pools virtuais

Nesses arquivos de definição de backend de exemplo, valores padrão específicos são definidos para todos os pools de storage, como `spaceReserve` em none, `spaceAllocation` em false e `encryption` em false. Os pools virtuais são definidos na seção de storage.

Trident define rótulos de provisionamento no campo "Comentários". Os comentários são definidos no volume FlexVol; Trident copia todos os rótulos presentes em um pool virtual para o volume de storage durante o provisionamento. Para conveniência, administradores de storage podem definir rótulos por pool virtual e agrupar volumes por rótulo.

Nestes exemplos, alguns pools de storage definem seus próprios `spaceReserve`, `spaceAllocation`, e `encryption` valores, e alguns pools substituem os valores padrão.

Exemplo de ONTAP SAN

```yaml
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "40000"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
      adaptiveQosPolicy: adaptive-extreme
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
      qosPolicy: premium
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

Exemplo de economia ONTAP SAN

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
labels:
  store: san_economy_store
region: us_east_1
storage:
  - labels:
      app: oracledb
      cost: "30"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
  - labels:
      app: postgresdb
      cost: "20"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
  - labels:
      app: mysqldb
      cost: "10"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
  - labels:
      department: legal
      creditpoints: "5000"
    zone: us_east_1c
```

```
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

Exemplo de NVMe/TCP

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

Mapear back-ends para StorageClasses

As seguintes definições de StorageClass referem-se ao Exemplos de backends com pools virtuais. Usando o campo `parameters.selector`, cada StorageClass especifica quais pools virtuais podem ser usados para hospedar um volume. O volume terá os aspectos definidos no pool virtual escolhido.

- O `protection-gold` StorageClass será mapeado para o primeiro pool virtual no `ontap-san` backend. Este é o único pool que oferece proteção de nível ouro.

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

- O `protection-not-gold` StorageClass corresponderá ao segundo e terceiro pool virtual no `ontap-san` backend. Esses são os únicos pools que oferecem um nível de proteção diferente de gold.

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

- O `app-mysqldb` StorageClass será mapeado para o terceiro pool virtual no `ontap-san-economy` backend. Este é o único pool que oferece configuração de pool de storage para o aplicativo do tipo mysqldb.

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

- O `protection-silver-creditpoints-20k` StorageClass será mapeado para o segundo pool virtual no `ontap-san` backend. Este é o único pool que oferece proteção de nível prata e 20000 pontos de crédito.

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

- O `creditpoints-5k` StorageClass corresponderá ao terceiro pool virtual no `ontap-san` backend e ao quarto pool virtual no `ontap-san-economy` backend. Essas são as únicas ofertas de pool com 5000 creditpoints.

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

- O `my-test-app-sc` StorageClass será mapeado para o `testAPP` pool virtual no `ontap-san` driver com `sanType: nvme`. Este é o único pool que oferece `testApp`.

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

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

Drivers NAS do ONTAP

Visão geral do driver ONTAP NAS

Saiba mais sobre como configurar um backend ONTAP com os drivers NAS do ONTAP e do Cloud Volumes ONTAP.

Detalhes do driver ONTAP NAS

Trident fornece os seguintes drivers de armazenamento NAS para se comunicar com o ONTAP cluster. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Driver	Protocolo	volumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`ontap-nas`	NFS SMB	Sistema de arquivos	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`
`ontap-nas-economy`	NFS SMB	Sistema de arquivos	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

Driver	Protocolo	volumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`ontap-nas-flexgroup`	NFS SMB	Sistema de arquivos	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

AVISO

- Use `ontap-san-economy` somente se a contagem de uso de volume persistente prevista for maior que "limites de volume ONTAP suportados".
- Use `ontap-nas-economy` somente se a contagem de uso de volume persistente prevista for maior que "limites de volume ONTAP suportados" e o driver `ontap-san-economy` não puder ser usado.
- Não utilize `ontap-nas-economy` se você prevê a necessidade de proteção de dados, recuperação de desastres ou mobilidade.
- NetApp não recomenda usar o crescimento automático do FlexVol em todos os drivers ONTAP, exceto ontap-san. Como alternativa, Trident oferece suporte ao uso de reserva de snapshot e dimensiona os volumes FlexVol de acordo.

Permissões do usuário

Trident espera ser executado como administrador do ONTAP ou do SVM, normalmente usando o `admin` usuário do cluster ou um `vsadmin` usuário do SVM, ou um usuário com um nome diferente que tenha a mesma função.

Para implantações do Amazon FSx for NetApp ONTAP, Trident espera ser executado como administrador do ONTAP ou do SVM, usando o usuário do cluster `fsxadmin` ou um usuário do SVM `vsadmin`, 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.

OBSERVAÇÃO

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

Embora seja possível criar uma função mais restritiva dentro do ONTAP que um driver Trident possa usar, não recomendamos isso. A maioria das novas versões do Trident chamará APIs adicionais que precisariam ser consideradas, dificultando as atualizações e tornando-as propensas a erros.

Prepare-se para configurar um backend com drivers ONTAP NAS

Compreenda os requisitos, as opções de autenticação e as políticas de exportação para configurar um backend ONTAP com drivers ONTAP NAS. A partir da versão 25.10, NetApp Trident oferece suporte "NetApp AFX sistema de storage". NetApp AFX storage systems diferem de outros sistemas ONTAP (ASA, AFF e FAS) na implementação de sua camada de storage. Na configuração do backend do Trident, não é necessário especificar que seu sistema é AFX. Ao selecionar `ontap-nas` como o `storageDriverName`, o Trident detecta automaticamente os sistemas AFX.

OBSERVAÇÃO	Apenas o `ontap-nas`driver (com o protocolo NFS) é compatível com sistemas AFX; o protocolo SMB não é compatível.

Requisitos

- Para todos os backends ONTAP, Trident exige que pelo menos um agregado seja atribuído à SVM.
- Você pode executar mais de um driver e criar classes de armazenamento que apontem para um ou outro. Por exemplo, você pode configurar uma classe Gold que usa o driver `ontap-nas` e uma classe Bronze que usa o `ontap-nas-economy`.
- Todos os seus nós de trabalho do Kubernetes devem ter as ferramentas NFS apropriadas instaladas. Consulte "aqui" para mais detalhes.
- Trident suporta volumes SMB montados em pods executados apenas em nós Windows. Consulte Prepare-se para provisionar volumes SMB para obter detalhes.

Autenticar o backend ONTAP

Trident oferece dois modos de autenticação de um backend ONTAP.

- Baseado em credenciais: Este modo requer permissões suficientes no backend do ONTAP. Recomenda-se usar uma conta associada a uma função de login de segurança predefinida, como `admin` ou `vsadmin` para garantir a máxima compatibilidade com as versões do ONTAP.
- Baseado em certificado: Este modo requer um certificado instalado no backend para o Trident se comunicar com um cluster ONTAP. Nesse caso, a definição do backend deve conter os valores codificados em Base64 do certificado do cliente, da chave e do certificado da CA confiável, se utilizado (recomendado).

Você pode atualizar os backends existentes para alternar entre métodos baseados em credenciais e baseados em certificados. No entanto, apenas um método de autenticação é suportado por vez. Para mudar para um método de autenticação diferente, você deve remover o método existente da configuração do backend.

AVISO	Se você tentar fornecer **tanto credenciais quanto certificados**, a criação do backend falhará com um erro informando que mais de um método de autenticação foi fornecido no arquivo de configuração.

Ativar autenticação baseada em credenciais

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

Uma definição de backend de exemplo será semelhante a esta:

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
        "name": "secret-backend-creds"
    }
}
```

Lembre-se de que a definição do backend é o único local onde as credenciais são armazenadas em texto simples. Após a criação do backend, nomes de usuário/senhas são codificados em Base64 e armazenados como segredos do Kubernetes. A criação/atualização de um backend é a única etapa que requer conhecimento das credenciais. Assim, trata-se de uma operação exclusiva do administrador, a ser realizada pelo administrador de storage do Kubernetes.

Habilitar autenticação baseada em certificado

Novos e existentes backends podem usar um certificado e se comunicar com o backend do ONTAP. Três parâmetros são necessários na definição do backend.

- clientCertificate: Valor codificado em Base64 do certificado do cliente.
- clientPrivateKey: Valor codificado em Base64 da chave privada associada.
- trustedCACertificate: valor codificado em Base64 do certificado CA confiável. Se uma CA confiável estiver sendo usada, este parâmetro deve ser fornecido. Isso pode ser ignorado se nenhuma CA confiável for usada.

Um fluxo de trabalho típico envolve as seguintes etapas.

Passos

1. Gere um certificado de cliente e uma chave. Ao gerar, defina o Nome Comum (CN) para o usuário ONTAP que será usado para autenticação.

```
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. Adicione um certificado de CA confiável ao cluster ONTAP. Isso pode já ter sido configurado pelo administrador de storage. Ignore se nenhuma CA confiável for utilizada.

```
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 do cliente e a chave (do passo 1) no cluster ONTAP.

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

4. Confirme se a função de login de segurança do ONTAP suporta `cert` 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 o certificado gerado. Substitua <ONTAP Management LIF> e <vserver name> pelo endereço IP da Management LIF e pelo 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. Codifique o certificado, a chave e o certificado da 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 o 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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Atualize os métodos de autenticação ou altere as credenciais

Você pode atualizar um backend existente para usar um método de autenticação diferente ou para rotacionar suas credenciais. Isso funciona nos dois sentidos: backends que utilizam nome de usuário/senha podem ser atualizados para usar certificados; backends que utilizam certificados podem ser atualizados para usar nome de usuário/senha. Para fazer isso, você deve remover o método de autenticação existente e adicionar o novo método de autenticação. Em seguida, use o arquivo backend.json atualizado contendo os parâmetros necessários para executar `tridentctl update backend`.

```
cat cert-backend-updated.json
```

```
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}
```

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

OBSERVAÇÃO	Ao rotacionar senhas, o administrador de storage deve primeiro atualizar a senha do usuário no ONTAP. Em seguida, é feita uma atualização no backend. Ao rotacionar certificados, vários certificados podem ser adicionados ao usuário. O backend é então atualizado para usar o novo certificado, após o que o certificado antigo pode ser excluído do cluster ONTAP.

A atualização do backend não interrompe o acesso aos volumes já criados, nem afeta as conexões de volume feitas posteriormente. Uma atualização bem-sucedida do backend indica que Trident pode se comunicar com o ONTAP backend e lidar com operações de volume futuras.

Criar função ONTAP personalizada para Trident

Você pode criar uma função de cluster ONTAP com privilégios mínimos para que não precise usar a função de administrador do ONTAP para executar operações no Trident. Ao incluir o nome de usuário em um arquivo de configuração de backend do Trident, o Trident usa a função de cluster ONTAP que você criou para executar as operações.

Consulte "Gerador de funções personalizadas Trident" para obter mais informações sobre como criar funções personalizadas do Trident.

Usando ONTAP CLI

1. Crie uma nova função usando o seguinte comando:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Crie um nome de usuário para o usuário do Trident:

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Mapeie a função para o usuário:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Usando System Manager

Execute as seguintes etapas no ONTAP System Manager:

1. **Crie uma função personalizada**:
   a. Para criar uma função personalizada no nível do cluster, selecione **Cluster > Settings**.

      (Ou) Para criar uma função personalizada no nível da SVM, selecione **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**.

   b. Selecione o ícone de seta (→) ao lado de **Users and Roles**.
   c. Selecione **+Adicionar** em **Roles**.
   d. Defina as regras para a função e clique em **Save**.
2. **Mapeie a função ao usuário Trident**: + Execute as seguintes etapas na página **Usuários e Funções**:
   a. Selecione o ícone Adicionar **+** em **Usuários**.
   b. Selecione o nome de usuário desejado e selecione uma função no menu suspenso para **Função**.
   c. Clique em **Salvar**.

Consulte as seguintes páginas para obter mais informações:

- "Funções personalizadas para administração do ONTAP" ou "Definir funções personalizadas"
- "Trabalhe com funções e usuários"

Gerenciar políticas de exportação NFS

Trident usa políticas de exportação NFS para controlar o acesso aos volumes que provisiona.

Trident oferece duas opções ao trabalhar com políticas de exportação:

- Trident pode gerenciar dinamicamente a própria política de exportação; nesse modo de operação, o

administrador de storage especifica uma lista de blocos CIDR que representam endereços IP admissíveis. Trident adiciona automaticamente os IPs de nó aplicáveis que se enquadram nesses intervalos à política de exportação no momento da publicação. Alternativamente, quando nenhum CIDR é especificado, todos os IPs unicast de escopo global encontrados no nó para o qual o volume está sendo publicado serão adicionados à política de exportação.

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

Gerencie dinamicamente as políticas de exportação

Trident oferece a capacidade de gerenciar dinamicamente políticas de exportação para backends ONTAP. Isso fornece ao administrador de storage a capacidade de especificar um espaço de endereços permitido para os endereços IP dos nós de trabalho, em vez de definir regras explícitas manualmente. Isso 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 storage apenas aos nós de trabalho que estão montando volumes e possuem endereços IP no intervalo especificado, oferecendo um gerenciamento detalhado e automatizado.

OBSERVAÇÃO	Não utilize Network Address Translation (NAT) ao usar políticas de exportação dinâmicas. Com NAT, o controlador de storage vê o endereço NAT de frontend e não o endereço IP real do host, portanto, o acesso será negado quando nenhuma correspondência for encontrada nas regras de exportação.

Exemplo

Existem duas opções de configuração que devem ser usadas. Veja um exemplo de definição de backend:

```
---
version: 1
storageDriverName: ontap-nas-economy
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
  - 192.168.0.0/24
autoExportPolicy: true
```

OBSERVAÇÃO	Ao usar este recurso, você deve garantir que a junção raiz em sua SVM tenha uma política de exportação previamente criada com uma regra de exportação que permita o bloco CIDR do nó (como a política de exportação padrão). Sempre siga a prática recomendada pela NetApp de dedicar uma SVM ao Trident.

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

- `autoExportPolicy` está definido como `true`. Isso indica que Trident cria uma política de exportação para cada volume provisionado com este backend para o `svm1` SVM e lida com a adição e exclusão de

regras usando `autoexportCIDRs` blocos de endereços. Até que um volume seja anexado a um nó, o volume usa uma política de exportação vazia, sem regras, para impedir o acesso indesejado a esse volume. Quando um volume é publicado em um nó, Trident cria uma política de exportação com o mesmo nome da qtree subjacente, contendo o endereço IP do nó dentro do bloco CIDR especificado. Esses IPs também serão adicionados à política de exportação usada pelo volume pai FlexVol.

  - Por exemplo:
    - UUID do backend 403b5326-8482-40db-96d0-d83fb3f4daec
    - `autoExportPolicy`definido para `true`
    - prefixo de storage `trident`
    - PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
    - qtree denominada trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c cria uma política de exportação para a FlexVol denominada `trident-403b5326-8482-40db96d0-d83fb3f4daec`, uma política de exportação para a qtree denominada `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`, e uma política de exportação vazia denominada `trident_empty` no SVM. As regras para a política de exportação da FlexVol serão um superconjunto de quaisquer regras contidas nas políticas de exportação da qtree. A política de exportação vazia será reutilizada por quaisquer volumes que não estejam anexados.

- `autoExportCIDRs` contém uma lista de blocos de endereços. Este campo é opcional e ele é definido por padrão como ["0.0.0.0/0", "::/0"]. Se não for definido, Trident adiciona todos os endereços unicast de escopo global encontrados nos nós de trabalho com publicações.

Neste exemplo, o `192.168.0.0/24` espaço de endereços é fornecido. Isso indica que os IPs dos nós do Kubernetes que se enquadram nesse intervalo de endereços com publicações serão adicionados à política de exportação que o Trident cria. Quando o Trident registra um nó no qual está sendo executado, ele recupera os endereços IP do nó e os compara com os blocos de endereços fornecidos em `autoExportCIDRs`. No momento da publicação, após filtrar os IPs, o Trident cria as regras da política de exportação para os endereços IP dos clientes do nó para o qual está publicando.

Você pode atualizar `autoExportPolicy` e `autoExportCIDRs` para backends após criá-los. Você pode adicionar novos CIDRs para um backend que é gerenciado automaticamente ou excluir CIDRs existentes. Tenha cuidado ao excluir CIDRs para garantir que conexões existentes não sejam interrompidas. Você também pode optar por desativar `autoExportPolicy` para um backend e voltar para uma política de exportação criada manualmente. Isso exigirá a configuração do parâmetro `exportPolicy` no seu arquivo de configuração do backend.

Após Trident criar ou atualizar um backend, você pode verificar o backend usando `tridentctl` ou o correspondente `tridentbackend` CRD:

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

Quando um nó é removido, Trident verifica todas as políticas de exportação para remover as regras de acesso correspondentes ao nó. Ao remover esse endereço IP do nó das políticas de exportação dos backends gerenciados, Trident impede montagens não autorizadas, a menos que esse endereço IP seja reutilizado por um novo nó no cluster.

Para backends preexistentes, atualizar o backend com `tridentctl update backend` garante que Trident gerencie as políticas de exportação automaticamente. Isso cria duas novas políticas de exportação nomeadas de acordo com o UUID do backend e o nome da qtree quando necessário. Volumes presentes no backend usarão as novas políticas de exportação após serem desmontados e montados novamente.

OBSERVAÇÃO	A exclusão de um backend 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, você deve reiniciar o pod do Trident nesse nó. Trident então atualizará a política de exportação dos backends que gerencia para refletir essa alteração de IP.

Prepare-se para provisionar volumes SMB

Com um pouco de preparação adicional, você pode provisionar volumes SMB usando `ontap-nas` drivers.

AVISO	Você deve configurar ambos os protocolos NFS e SMB/CIFS na SVM para criar um `ontap-nas-economy` volume SMB para clusters ONTAP locais. A falha ao configurar qualquer um desses protocolos fará com que a criação do volume SMB falhe.

OBSERVAÇÃO `autoExportPolicy` não é compatível com volumes SMB.

Antes de começar

Antes de poder provisionar volumes SMB, você deve ter o seguinte.

- Um cluster Kubernetes com um nó controlador Linux e pelo menos um nó de trabalho Windows executando Windows Server 2022. Trident suporta volumes SMB montados em pods executados apenas em nós Windows.
- Pelo menos um segredo Trident contendo suas credenciais do Active Directory. Para gerar o segredo `smbcreds`:

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

- Um proxy CSI configurado como um serviço do Windows. Para configurar um `csi-proxy`, consulte "GitHub: CSI Proxy" ou "GitHub: CSI Proxy para Windows" para nós do Kubernetes em execução no Windows.

Passos

1. Para o ONTAP local, você pode opcionalmente criar um compartilhamento SMB ou o Trident pode criar um para você.

   OBSERVAÇÃO Os compartilhamentos SMB são necessários para Amazon FSx for ONTAP.

   Você pode criar os compartilhamentos administrativos SMB de duas maneiras: usando o "Microsoft Management Console" snap-in Shared Folders ou usando a ONTAP CLI. Para criar os compartilhamentos SMB usando a ONTAP CLI:

   a. Se necessário, crie a estrutura de caminho de diretórios para o compartilhamento.

      O `vserver cifs share create` comando verifica o caminho especificado na opção -path durante a criação do compartilhamento. Se o caminho especificado não existir, o comando falha.

   b. Crie um compartilhamento SMB associado à SVM especificada:

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

   c. Verifique se o compartilhamento foi criado:

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

   OBSERVAÇÃO Consulte "Criar um compartilhamento SMB" para obter detalhes completos.

2. Ao criar o backend, você deve configurar o seguinte para especificar os volumes SMB. Para todas as

opções de configuração do backend do FSx para ONTAP, consulte "Opções e exemplos de configuração do FSx for ONTAP".

Parâmetro	Descrição	Exemplo
`smbShare`	Você pode especificar uma das seguintes opções: o nome de um compartilhamento SMB criado usando o Microsoft Management Console ou ONTAP CLI; um nome para permitir que o Trident crie o compartilhamento SMB; ou você pode deixar o parâmetro em branco para impedir o acesso comum aos volumes. Este parâmetro é opcional para ONTAP on-premises. Este parâmetro é obrigatório para Amazon FSx for ONTAP backends e não pode ficar em branco.	`smb-share`
`nasType`	**Deve ser definido como `smb`.** Se for nulo, o padrão é `nfs`.	`smb`
`securityStyle`	Estilo de segurança para novos volumes. **Deve ser definido como `ntfs` ou `mixed` para volumes SMB.**	`ntfs` ou `mixed` for SMB volumes
`unixPermissions`	Modo para novos volumes. **Deve ser deixado em branco para volumes SMB.**	""

Ativar SMB seguro

A partir da versão 25.06, NetApp Trident oferece suporte ao provisionamento seguro de volumes SMB criados usando `ontap-nas` e `ontap-nas-economy` backends. Quando o SMB seguro está habilitado, você pode fornecer acesso controlado aos compartilhamentos SMB para usuários e grupos de usuários do Active Directory (AD) usando listas de controle de acesso (ACLs).

Pontos a lembrar

- A importação de `ontap-nas-economy` volumes não é suportada.
- Apenas clones somente leitura são suportados para `ontap-nas-economy` volumes.
- Se o Secure SMB estiver ativado, Trident ignorará o compartilhamento SMB mencionado no backend.
- A atualização da anotação PVC, da anotação da storage class e do campo backend não atualiza a ACL de compartilhamento SMB.
- A ACL de compartilhamento SMB especificada na anotação do PVC clonado terá precedência sobre as do PVC de origem.
- Certifique-se de fornecer usuários válidos do Active Directory ao habilitar o SMB seguro. Usuários inválidos não serão adicionados à ACL.
- Se você fornecer o mesmo usuário do AD no backend, na storage class e no PVC com permissões diferentes, a prioridade de permissão será: PVC, storage class e, em seguida, backend.
- Secure SMB é compatível com `ontap-nas`importações de volumes gerenciados e não se aplica a importações de volumes não gerenciados.

Passos

1. Especifique adAdminUser em TridentBackendConfig conforme mostrado no exemplo a seguir:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. Adicione a anotação na storage class.

   Adicione a `trident.netapp.io/smbShareAdUser` anotação à storage class para habilitar SMB seguro sem falhas. O valor de usuário especificado para a anotação `trident.netapp.io/smbShareAdUser` deve ser o mesmo que o nome de usuário especificado no `smbcreds` secret. Você pode escolher um dos seguintes para `smbShareAdUserPermission`: `full_control`, `change` ou `read`. A permissão padrão é `full_control`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1. Crie um PVC.

   O exemplo a seguir cria um PVC:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

Opções e exemplos de configuração do ONTAP NAS

Aprenda a criar e usar drivers ONTAP NAS com sua instalação do Trident. Esta seção fornece exemplos de configuração de backend e detalhes para mapear backends para StorageClasses. A partir da versão 25.10, NetApp Trident oferece suporte "NetApp sistemas de storage AFX". NetApp AFX sistemas de storage diferem de outros sistemas baseados em ONTAP (ASA, AFF e FAS) na implementação de sua camada de storage.

OBSERVAÇÃO	Apenas o `ontap-nas`driver (com protocolo NFS) é compatível com sistemas AFX da NetApp; o protocolo SMB não é compatível.

Opções de configuração do backend

Na configuração do backend do Trident, não é necessário especificar que seu sistema é um NetApp AFX sistema de storage. Ao selecionar `ontap-nas` como o `storageDriverName`, o Trident detecta automaticamente o sistema de storage AFX. Alguns parâmetros de configuração do backend não se aplicam a sistemas de storage AFX.

A tabela a seguir exibe as opções de configuração do backend:

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do driver de armazenamento **OBSERVAÇÃO** Para sistemas NetApp AFX, apenas `ontap-nas` é suportado.	`ontap-nas`, `ontap-nas-economy`, ou `ontap-nas-flexgroup`

Parâmetro	Descrição	Padrão
`backendName`	Nome personalizado ou o storage backend	Nome do driver + "_" + dataLIF
`managementLIF`	Endereço IP de um cluster ou LIF de gerenciamento de SVM. Um nome de domínio totalmente qualificado (FQDN) pode ser especificado. Pode ser configurado para usar endereços IPv6 se Trident foi instalado com o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Para um switchover MetroCluster perfeito, consulte o Exemplo do MetroCluster.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Endereço IP do protocolo LIF. NetApp recomenda especificar `dataLIF`. Caso não seja fornecido, Trident busca os dataLIFs da SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) para ser usado nas operações de montagem NFS, permitindo criar um DNS round-robin para balancear a carga entre vários dataLIFs. Pode ser alterado após a configuração inicial. Consulte . Pode ser configurado para usar endereços IPv6 se Trident foi instalado com o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **Omita para MetroCluster.** Consulte o Exemplo do MetroCluster.	Endereço especificado ou derivado de SVM, caso não seja especificado (não recomendado)
`svm`	Máquina virtual de storage a ser usada **Omitir para MetroCluster.** Consulte o Exemplo do MetroCluster.	Derivado se uma SVM `managementLIF` for especificada
`autoExportPolicy`	Ativar a criação e atualização automática de políticas de exportação [Booleano]. Usando as opções `autoExportPolicy` e `autoExportCIDRs`, Trident pode gerenciar políticas de exportação automaticamente.	falso
`autoExportCIDRs`	Lista de CIDRs para filtrar os IPs dos nós do Kubernetes quando `autoExportPolicy` estiver habilitado. Usando as opções `autoExportPolicy` e `autoExportCIDRs`, Trident pode gerenciar políticas de exportação automaticamente.	["0.0.0.0/0", "::/0"]`
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar aos volumes	""
`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	""

Parâmetro	Descrição	Padrão
`trustedCACertificate`	Valor codificado em Base64 do certificado da CA confiável. Opcional. Usado para autenticação baseada em certificado	""
`username`	Nome de usuário para conectar-se ao cluster/SVM. Usada para autenticação baseada em credenciais. Para autenticação do Active Directory, consulte "Autentique o Trident em um SVM de backend usando credenciais do Active Directory".	
`password`	Senha para conectar-se ao cluster/SVM. Usada para autenticação baseada em credenciais. Para autenticação do Active Directory, consulte "Autentique o Trident em um SVM de backend usando credenciais do Active Directory".	
`storagePrefix`	Prefixo usado ao provisionar novos volumes no SVM. Não pode ser atualizado após ser definido **OBSERVAÇÃO** Ao usar o ontap-nas-economy e um storagePrefix com 24 ou mais caracteres, as qtrees não terão o storage prefix incorporado, embora ele esteja presente no nome do volume.	"Trident"

Parâmetro	Descrição	Padrão
`aggregate`	Agregado para provisionamento (opcional; se definido, deve ser atribuído à SVM). Para o `ontap-nas-flexgroup` driver, esta opção é ignorada. Se não for atribuído, qualquer um dos agregados disponíveis pode ser usado para provisionar um FlexGroup volume. **OBSERVAÇÃO**: Quando o agregado é atualizado no SVM, ele é atualizado automaticamente no Trident por meio de polling no SVM, sem a necessidade de reiniciar o Trident Controller. Quando você configurou um agregado específico no Trident para provisionar volumes, se o agregado for renomeado ou movido para fora do SVM, o backend entrará em estado de falha no Trident durante o polling do agregado no SVM. Você deve alterar o agregado para um que esteja presente no SVM ou removê-lo completamente para que o backend volte a ficar online. **Não especifique para sistemas de storage AFX.**	""
`limitAggregateUsage`	O provisionamento falhará se o uso ultrapassar essa porcentagem. **Não se aplica ao Amazon FSx para ONTAP**. **Não especifique para sistemas de storage AFX.**	"" (não aplicado por padrão)

Parâmetro	Descrição	Padrão
flexgroupAggregateList	Lista de agregados para provisionamento (opcional; se definida, deve ser atribuída à SVM). Todos os agregados atribuídos à SVM são usados para provisionar um FlexGroup volume. Compatível com o driver de storage **ontap-nas-flexgroup**. **OBSERVAÇÃO**: Quando a lista de agregados é atualizada no SVM, a lista é atualizada automaticamente no Trident por meio de polling do SVM, sem a necessidade de reiniciar o Trident Controller. Quando você configurou uma lista de agregados específica no Trident para provisionar volumes, se a lista de agregados for renomeada ou movida para fora do SVM, o backend entrará em estado de falha no Trident enquanto faz polling do agregado do SVM. Você deve alterar a lista de agregados para uma que esteja presente no SVM ou removê-la completamente para que o backend volte a ficar online.	""
`limitVolumeSize`	O provisionamento falha se o tamanho do volume solicitado for superior a este valor.	"" (não aplicado por padrão)
`debugTraceFlags`	Sinalizadores de depuração para usar na resolução de problemas. Exemplo, {"api":false, "method":true} não use `debugTraceFlags` a menos que esteja solucionando problemas e precise de um despejo de log detalhado.	null
`nasType`	Configurar a criação de volumes NFS ou SMB. As opções são `nfs`, `smb` ou null. Definir como null define volumes NFS por padrão. **Se especificado, defina sempre como `nfs` para sistemas de armazenamento AFX**.	`nfs`

Parâmetro	Descrição	Padrão
`nfsMountOptions`	Lista separada por vírgulas de opções de montagem NFS. As opções de montagem para volumes persistentes do Kubernetes são normalmente especificadas nas classes de armazenamento, mas se nenhuma opção de montagem for especificada em uma classe de armazenamento, Trident usará as opções de montagem especificadas no arquivo de configuração do backend de storage. Se nenhuma opção de montagem for especificada na classe de armazenamento ou no arquivo de configuração, Trident não definirá nenhuma opção de montagem em um volume persistente associado.	""
`qtreesPerFlexvol`	Máximo de Qtrees por FlexVol, deve estar no intervalo [50, 300]	"200"
`smbShare`	Você pode especificar uma das seguintes opções: o nome de um compartilhamento SMB criado usando o Microsoft Management Console ou ONTAP CLI; um nome para permitir que o Trident crie o compartilhamento SMB; ou você pode deixar o parâmetro em branco para impedir o acesso comum aos volumes. Este parâmetro é opcional para ONTAP on-premises. Este parâmetro é obrigatório para Amazon FSx for ONTAP backends e não pode ficar em branco.	`smb-share`
`useREST`	Parâmetro booleano para usar as ONTAP REST APIs. `useREST` Quando definido como `true`, Trident usa as ONTAP REST APIs para se comunicar com o backend; quando definido como `false`, Trident usa chamadas ONTAPI (ZAPI) para se comunicar com o backend. Este recurso requer ONTAP 9.11.1 e versões posteriores. Além disso, a função de login do ONTAP utilizada deve ter acesso ao aplicativo `ontapi`. Isso é atendido pelas funções predefinidas `vsadmin` e `cluster-admin`. A partir da versão 24.06 do Trident e ONTAP 9.15.1 ou posterior, `useREST` é definido como `true` por padrão; altere `useREST` para `false` para usar chamadas ONTAPI (ZAPI). **Se especificado, defina sempre como `true` para sistemas de armazenamento AFX**.	`true`para ONTAP 9.15.1 ou posterior, caso contrário `false`.
`limitVolumePoolSize`	Tamanho máximo solicitável de FlexVol ao usar Qtrees no backend ontap-nas-economy.	"" (não aplicado por padrão)
`denyNewVolumePools`	Restringe `ontap-nas-economy` os backends de criarem novos volumes FlexVol para conter seus Qtrees. Somente FlexVols preexistentes são usados para provisionar novos PVs.	

Parâmetro	Descrição	Padrão
`adAdminUser`	Usuário ou grupo de usuários administradores do Active Directory com acesso total aos compartilhamentos SMB. Use este parâmetro para conceder direitos de administrador ao compartilhamento SMB com controle total.	

Opções de configuração de backend para provisionamento de volumes

Você pode controlar o provisionamento padrão usando essas opções na seção `defaults` do arquivo de configuração. Para um exemplo, veja os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
`spaceAllocation`	Alocação de espaço para Qtrees	"true"
`spaceReserve`	Modo de reserva de espaço; "none" (fino) ou "volume" (grosso)	"none"
`snapshotPolicy`	Política do Snapshot a ser usada	"none"
`qosPolicy`	Grupo de políticas de QoS a ser atribuído aos volumes criados. Escolha uma de qosPolicy ou adaptiveQosPolicy por pool de storage/backend	""
`adaptiveQosPolicy`	Grupo de políticas de QoS adaptável para atribuir aos volumes criados. Escolha um dos qosPolicy ou adaptiveQosPolicy por pool de storage/backend. Não compatível com ontap-nas-economy.	""
`snapshotReserve`	Percentual do volume reservado para snapshots	"0" se `snapshotPolicy` for "none", caso contrário ""
`splitOnClone`	Separar um clone de seu progenitor no momento da criação	"false"
`encryption`	Habilite NetApp Volume Encryption (NVE) no novo volume; o padrão é `false`. A NVE deve estar licenciada e habilitada no cluster para usar esta opção. Se a NAE estiver habilitada no backend, qualquer volume provisionado no Trident terá a NAE habilitada. Para mais informações, consulte: "Como Trident funciona com NVE e NAE".	"false"
`tieringPolicy`	Política de tiering para usar "none"	
`unixPermissions`	Modo para novos volumes	"777" para volumes NFS; vazio (não aplicável) para volumes SMB
`snapshotDir`	Controla o acesso ao `.snapshot` diretório	`true`, `false` (Definido explicitamente).
`exportPolicy`	Política de exportação a ser usada	"default"

Parâmetro	Descrição	Padrão
`securityStyle`	Estilo de segurança para novos volumes. NFS suporta `mixed` e `unix` estilos de segurança. SMB suporta `mixed` e `ntfs` estilos de segurança.	NFS default é `unix`. SMB default é `ntfs`.
`nameTemplate`	Modelo para criar nomes de volume personalizados.	""

OBSERVAÇÃO	O uso de grupos de políticas de QoS com Trident requer ONTAP 9.8 ou posterior. Você deve usar um grupo de políticas de QoS não compartilhado e garantir que o grupo de políticas seja aplicado a cada componente individualmente. Um grupo de políticas de QoS compartilhado impõe o limite máximo para a taxa de transferência total de todas as cargas de trabalho.

Exemplos de provisionamento de volume

Aqui está um exemplo com valores padrão 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 Trident agora usa um novo cálculo para garantir que o FlexVol seja dimensionado corretamente com a porcentagem de snapshotReserve e o PVC. Quando o usuário solicita um PVC, o Trident cria o FlexVol original com mais espaço usando o novo cálculo. Esse cálculo garante que o usuário receba o espaço gravável solicitado no PVC, e não menos espaço do que o solicitado. Antes da v21.07, quando o usuário solicitava um PVC (por exemplo, 5 GiB), com o

snapshotReserve em 50 por cento, ele recebia apenas 2,5 GiB de espaço gravável. Isso ocorre porque o que o usuário solicitava era o volume total e `snapshotReserve` é uma porcentagem disso. Com o Trident 21.07, o que o usuário solicita é o espaço gravável e o Trident define o número de `snapshotReserve` como a porcentagem do volume total. Isso não se aplica a `ontap-nas-economy`. 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 snapshotReserve = 50%, e solicitação de PVC = 5 GiB, o tamanho total do volume é 5/0,5 = 10 GiB e o tamanho disponível é 5 GiB, que é o que o usuário solicitou na solicitação de PVC. O comando `volume show` deve exibir resultados semelhantes a este exemplo:

Os backends existentes de instalações anteriores provisionarão volumes conforme explicado acima ao atualizar 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 2 GiB com `snapshotReserve=50` anteriormente resultava em um volume que fornece 1 GiB de espaço gravável. Redimensionar o volume para 3 GiB, por exemplo, fornece ao aplicativo 3 GiB de espaço gravável em um volume de 6 GiB.

Exemplos de configuração mínima

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

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

Exemplo de economia ONTAP NAS

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

Exemplo de FlexGroup ONTAP NAS

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

Exemplo do MetroCluster

Você pode configurar o backend para evitar ter que atualizar manualmente a definição do backend após switchover e switchback durante "Replicação e recuperação de SVM".

Para switchover e switchback sem interrupções, especifique a SVM usando `managementLIF` e omita os parâmetros `dataLIF` e `svm`. Por exemplo:

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

Exemplo de volumes SMB

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

Exemplo de autenticação baseada em certificado

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

```
---
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_
```

Exemplo de política de exportação automática

Este exemplo mostra como você pode instruir 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` e `ontap-nas-flexgroup`drivers.

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

Exemplo de endereços IPv6

Este exemplo mostra `managementLIF` usando um endereço IPv6.

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

Amazon FSx para ONTAP usando exemplo de volumes SMB

O `smbShare` parâmetro é necessário para Amazon FSx para ONTAP usando volumes SMB.

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

Exemplo de configuração de backend com nameTemplate

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap-nas-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

Exemplos de backends com pools virtuais

Nos arquivos de definição de backend de exemplo mostrados abaixo, valores padrão específicos são definidos para todos os pools de storage, como `spaceReserve` em none, `spaceAllocation` em false e `encryption` em false. Os pools virtuais são definidos na seção de storage.

Trident define rótulos de provisionamento no campo "Comentários". Os comentários são definidos em FlexVol para `ontap-nas` ou FlexGroup para `ontap-nas-flexgroup`. Trident copia todos os rótulos presentes em um pool virtual para o volume de armazenamento durante o provisionamento. Para conveniência, administradores de storage podem definir rótulos por pool virtual e agrupar volumes por rótulo.

Nestes exemplos, alguns pools de storage definem seus próprios `spaceReserve`, `spaceAllocation`, e `encryption` valores, e alguns pools substituem os valores padrão.

Exemplo de ONTAP NAS

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
nfsMountOptions: nfsvers=4
defaults:
  spaceReserve: none
  encryption: "false"
  qosPolicy: standard
labels:
  store: nas_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      app: msoffice
      cost: "100"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
      adaptiveQosPolicy: adaptive-premium
  - labels:
      app: slack
      cost: "75"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: legal
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      app: wordpress
```

```
      cost: "50"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      app: mysqldb
      cost: "25"
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

Exemplo de ONTAP NAS FlexGroup

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: flexgroup_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "50000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: gold
      creditpoints: "30000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      protection: bronze
      creditpoints: "10000"
    zone: us_east_1d
    defaults:
```

```
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

Exemplo de economia ONTAP NAS

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: nas_economy_store
region: us_east_1
storage:
  - labels:
      department: finance
      creditpoints: "6000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: engineering
      creditpoints: "3000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      department: humanresource
      creditpoints: "2000"
    zone: us_east_1d
    defaults:
      spaceReserve: volume
```

```
      encryption: "false"
      unixPermissions: "0775"
```

Mapear back-ends para StorageClasses

As seguintes definições de StorageClass referem-se a Exemplos de backends com pools virtuais. Usando o campo `parameters.selector`, cada StorageClass especifica quais pools virtuais podem ser usados para hospedar um volume. O volume terá os aspectos definidos no pool virtual escolhido.

- O `protection-gold` StorageClass corresponderá ao primeiro e ao segundo pool virtual no `ontap-nas-flexgroup` backend. Esses são os únicos pools que oferecem proteção de nível ouro.

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

- O `protection-not-gold` StorageClass corresponderá ao terceiro e quarto pool virtual no `ontap-nas-flexgroup` backend. Esses são os únicos pools que oferecem nível de proteção diferente de gold.

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

- O `app-mysqldb` StorageClass será mapeado para o quarto pool virtual no `ontap-nas` backend. Este é o único pool que oferece configuração de pool de storage para app do tipo mysqldb.

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

- O `protection-silver-creditpoints-20k` StorageClass será mapeado para o terceiro pool virtual no `ontap-nas-flexgroup` backend. Este é o único pool que oferece proteção de nível prata e 20000 pontos de crédito.

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

- O `creditpoints-5k` StorageClass corresponderá ao terceiro pool virtual no `ontap-nas` backend e ao segundo pool virtual no `ontap-nas-economy` backend. Essas são as únicas ofertas de pool com 5000 creditpoints.

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

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

Atualize `dataLIF` após a configuração inicial

Você pode alterar o dataLIF após a configuração inicial executando o seguinte comando para fornecer o novo arquivo JSON de backend com o dataLIF atualizado.

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

OBSERVAÇÃO	Se os PVCs estiverem conectados a um ou mais pods, você deve desligar todos os pods correspondentes e então ligá-los novamente para que o novo dataLIF entre em vigor.

Exemplos seguros de SMB

Configuração de backend com o driver ontap-nas

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Configuração de backend com o driver ontap-nas-economy

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Configuração de backend com pool de storage

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Exemplo de classe de armazenamento com o driver ontap-nas

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

OBSERVAÇÃO	Certifique-se de adicionar `annotations` para habilitar o SMB seguro. O SMB seguro não funciona sem as anotações, independentemente das configurações definidas no Backend ou no PVC.

Exemplo de classe de armazenamento com o driver ontap-nas-economy

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

Exemplo de PVC com um único usuário AD

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

Exemplo de PVC com vários usuários de AD

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Amazon FSx for NetApp ONTAP

Use Trident com Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP" é um serviço totalmente gerenciado da AWS que executa sistemas de arquivos alimentados pelo sistema operacional de storage NetApp ONTAP. Ele fornece recursos, desempenho e administração do ONTAP com a escalabilidade e simplicidade operacional da AWS. Um sistema de arquivos é o recurso principal no Amazon FSx e é análogo a um cluster ONTAP local. Cada sistema de arquivos contém uma ou mais máquinas virtuais de armazenamento (SVMs), e cada SVM contém um ou mais volumes que armazenam arquivos e diretórios. Essa integração permite que clusters Kubernetes em execução no Amazon Elastic Kubernetes Service (EKS) provisionem volumes persistentes com suporte a ONTAP para cargas de trabalho de bloco e arquivo.

Requisitos

Além de "Requisitos do Trident", para integrar o FSx for ONTAP com Trident, você precisa de:

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

OBSERVAÇÃO	Certifique-se de seguir os passos de preparação do nó necessários para Amazon Linux e Ubuntu "Amazon Machine Images" (AMIs), dependendo do seu tipo de AMI do EKS.

Considerações

- Volumes SMB:
  - Os volumes SMB são suportados usando apenas o driver `ontap-nas`.
  - Os volumes SMB não são suportados com o Trident EKS add-on.
  - Trident suporta volumes SMB montados em pods executados apenas em nós Windows. Consulte "Prepare-se para provisionar volumes SMB" para obter detalhes.
- Antes do Trident 24.02, volumes criados em sistemas de arquivos Amazon FSx que tinham backups automáticos ativados não podiam ser excluídos pelo Trident. Para evitar esse problema no Trident 24.02 ou posterior, especifique o `fsxFilesystemID`, AWS `apiRegion`, AWS `apikey` e AWS `secretKey` no arquivo de configuração do backend para AWS FSx for ONTAP.

OBSERVAÇÃO	Se você estiver especificando uma função do IAM para Trident, poderá omitir a especificação dos campos `apiRegion`, `apiKey` e `secretKey` para o Trident explicitamente. Para obter mais informações, consulte "Opções e exemplos de configuração do FSx for ONTAP".

Uso simultâneo do Trident SAN/iSCSI e do driver EBS-CSI

Se você planeja usar drivers ontap-san (por exemplo, iSCSI) com AWS (EKS, ROSA, EC2 ou qualquer outra instância), a configuração de multipath necessária nos nós pode entrar em conflito com o driver CSI do Amazon Elastic Block Store (EBS). Para garantir que o multipath funcione sem interferir nos discos EBS no mesmo nó, você precisa excluir EBS da sua configuração de multipath. Este exemplo mostra um `multipath.conf`arquivo que inclui as configurações necessárias do Trident, excluindo os discos EBS do multipath:

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

Autenticação

Trident oferece dois modos de autenticação.

- Baseado em credenciais (recomendado): armazena as credenciais com segurança no AWS Secrets Manager. Você pode usar o `fsxadmin` usuário do seu sistema de arquivos ou o `vsadmin` usuário configurado para sua SVM.

> **AVISO**
>
> Trident espera ser executado como um `vsadmin` usuário SVM ou como um usuário com um nome diferente que tenha a mesma função. Amazon FSx for NetApp ONTAP possui um `fsxadmin` usuário que é um substituto limitado para o ONTAP `admin` cluster user. Recomendamos fortemente usar `vsadmin` com Trident.

- Com base em certificado: Trident se comunicará com a SVM no seu sistema de arquivos FSx usando um certificado instalado na sua SVM.

Para obter detalhes sobre como ativar a autenticação, consulte a autenticação para o seu tipo de driver:

- "ONTAP NAS autenticação"
- "ONTAP SAN autenticação"

Imagens de Máquina da Amazon (AMIs) testadas

O cluster EKS suporta diversos sistemas operacionais, mas a AWS otimizou determinadas Amazon Machine Images (AMIs) para contêineres e EKS. As seguintes AMIs foram testadas com NetApp Trident 25.02.

AMI	NAS	NAS-economy	iSCSI	iSCSI-economia
AL2023_x86_64_STANDARD	Sim	Sim	Sim	Sim
AL2_x86_64	Sim	Sim	Sim*	Sim*
BOTTLEROCKET_x86_64	Sim**	Sim	N/A	N/A
AL2023_ARM_64_STANDARD	Sim	Sim	Sim	Sim
AL2_ARM_64	Sim	Sim	Sim*	Sim*

BOTTLEROCKET_ARM_64	Sim**	Sim	N/A	N/A

- * Não foi possível excluir o PV sem reiniciar o nó
- ** Não funciona com NFSv3 com Trident versão 25.02.

OBSERVAÇÃO	Se a AMI desejada não estiver listada aqui, isso não significa que ela não seja compatível; significa apenas que ela não foi testada. Esta lista serve como um guia para AMIs que comprovadamente funcionam.

Testes realizados com:

- Versão do EKS: 1.32
- Método de instalação: Helm 25.06 e como um AWS add-On 25.06
- Para NAS, foram testados tanto NFSv3 quanto NFSv4.1.
- Para SAN, apenas iSCSI foi testado, não NVMe-oF.

Testes realizados:

- Criar: classe de armazenamento, pvc, pod
- Excluir: pod, pvc (regular, qtree/lun – econômico, NAS com backup da AWS)

Encontre mais informações

- "Documentação do Amazon FSx for NetApp ONTAP"
- "Postagem no blog sobre Amazon FSx for NetApp ONTAP"

Crie uma função do IAM e um segredo da AWS

Você pode configurar pods do Kubernetes para acessar recursos da AWS autenticando-se como uma função do AWS IAM em vez de fornecer credenciais explícitas da AWS.

OBSERVAÇÃO	Para autenticar usando uma função do AWS IAM, você deve ter um cluster Kubernetes implantado usando EKS.

Criar segredo do AWS Secrets Manager

Como Trident emitirá APIs contra um FSx vserver para gerenciar o storage para você, ele precisará de credenciais para isso. A maneira segura de passar essas credenciais é por meio de um segredo do AWS Secrets Manager. Portanto, se você ainda não tiver um, precisará criar um segredo do AWS Secrets Manager que contenha as credenciais da conta vsadmin.

Este exemplo cria um segredo do AWS Secrets Manager para armazenar credenciais do Trident CSI:

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

Criar política de IAM

Trident também precisa de permissões da AWS para funcionar corretamente. Portanto, você precisa criar uma política que conceda ao Trident as permissões de que ele precisa.

Os exemplos a seguir criam uma política do IAM usando a AWS CLI:

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy
-document file://policy.json
  --description "This policy grants access to Trident CSI to FSxN and
Secrets manager"
```

Exemplo de JSON de policy:

```
{
  "Statement": [
    {
      "Action": [
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes",
        "fsx:CreateVolume",
        "fsx:RestoreVolumeFromSnapshot",
        "fsx:DescribeStorageVirtualMachines",
        "fsx:UntagResource",
        "fsx:UpdateVolume",
        "fsx:TagResource",
        "fsx:DeleteVolume"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "secretsmanager:GetSecretValue",
      "Effect": "Allow",
      "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-
id>:secret:<aws-secret-manager-name>*"
    }
  ],
  "Version": "2012-10-17"
}
```

Criar identidade de Pod ou função IAM para associação de conta de serviço (IRSA)

Você pode configurar uma conta de serviço do Kubernetes para assumir uma função do AWS Identity and Access Management (IAM) com EKS Pod Identity ou IAM role for Service account association (IRSA). Quaisquer Pods configurados para usar a conta de serviço podem então acessar qualquer serviço da AWS ao

qual a função tenha permissões de acesso.

Identidade do Pod

As associações de identidade de pods do Amazon EKS permitem gerenciar credenciais para seus aplicativos, de forma semelhante à maneira como os perfis de instância do Amazon EC2 fornecem credenciais para instâncias do Amazon EC2.

Instale o Pod Identity no seu cluster EKS:

Você pode criar uma identidade de Pod via console da AWS ou usando o seguinte comando da AWS CLI:

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

Para obter mais informações, consulte "Configure o agente de identidade do Amazon EKS Pod".

Crie trust-relationship.json:

Crie o arquivo trust-relationship.json para permitir que o EKS Service Principal assuma essa função para Pod Identity. Em seguida, crie uma função com esta trust policy:

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

arquivo trust-relationship.json:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Anexe a política de função à função do IAM:

Anexe a política de função da etapa anterior à função IAM que foi criada:

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

Crie uma associação de identidade de pod:

Crie uma associação de identidade de pod entre a função IAM e a conta de serviço Trident (trident-controller)

```
aws eks create-pod-identity-association \
  --cluster-name <EKS_CLUSTER_NAME> \
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \
  --namespace trident --service-account trident-controller
```

Função IAM para associação de conta de serviço (IRSA)

Usando a AWS CLI:

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

Arquivo trust-relationship.json

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

Atualize os seguintes valores no arquivo `trust-relationship.json`:

- **<account_id>** - Seu ID de conta da AWS
- **<oidc_provider>** - O OIDC do seu cluster EKS. Você pode obter o oidc_provider executando:

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

Associe a função IAM à política IAM:

Depois que a função for criada, associe a política (que foi criada na etapa acima) à função usando este comando:

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

Verifique se o provedor do OICD está associado:

Verifique se o seu provedor OIDC está associado ao seu cluster. Você pode verificar isso usando este comando:

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Se a saída estiver vazia, use o seguinte comando para associar IAM OIDC ao seu cluster:

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

Se você estiver usando o eksctl, use o exemplo a seguir para criar uma função IAM para conta de serviço no EKS:

```
eksctl create iamserviceaccount --name trident-controller --namespace
trident \
  --cluster <my-cluster> --role-name AmazonEKS_FSxN_CSI_DriverRole
--role-only \
  --attach-policy-arn <IAM-Policy ARN> --approve
```

Instale Trident

Trident simplifica o gerenciamento de storage do Amazon FSx for NetApp ONTAP no Kubernetes para permitir que seus desenvolvedores e administradores se concentrem na implantação de aplicativos. Você pode instalar Trident usando um dos seguintes métodos:

- Helm
- Complemento do EKS

Se você deseja utilizar a funcionalidade de instantâneo, instale o complemento do controlador de instantâneo CSI. Consulte "Ative a funcionalidade de instantâneo para volumes CSI" para mais informações.

Instale Trident via helm

Identidade do Pod

1. Adicione o repositório Trident:

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

2. Instale Trident usando o seguinte exemplo:

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 --namespace trident --create-namespace
```

Você pode usar o `helm list` comando para revisar detalhes da instalação, como nome, namespace, chart, status, versão do aplicativo e número da revisão.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                           APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2502.0    25.02.0
```

Associação de conta de serviço (IRSA)

1. Adicione o repositório Trident:

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

2. Defina os valores para **provedor de nuvem** e **identidade de nuvem**:

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

Você pode usar o `helm list` comando para revisar detalhes da instalação, como nome, namespace, chart, status, versão do aplicativo e número da revisão.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                           APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2510.0    25.10.0
```

OBSERVAÇÃO

Se você pretende usar iSCSI, certifique-se de que o iSCSI esteja habilitado em sua máquina cliente. Se você estiver usando o sistema operacional AL2023 Worker node, é possível automatizar a instalação do cliente iSCSI adicionando o parâmetro node prep na instalação do helm:

```
helm install trident-operator netapp-trident/trident-
operator --version 100.2502.1 --namespace trident
--create-namespace --set nodePrep={iscsi}
```

Instale o Trident via o add-on EKS

O Trident EKS add-on inclui os patches de segurança mais recentes, correções de bugs e é validado pela AWS para funcionar com o Amazon EKS. O EKS add-on permite garantir de forma consistente que seus clusters Amazon EKS estejam seguros e estáveis e reduz a quantidade de trabalho necessária para instalar, configurar e atualizar add-ons.

Pré-requisitos

Certifique-se de ter o seguinte antes de configurar o add-on Trident para AWS EKS:

- Uma conta de cluster Amazon EKS com assinatura adicional
- Permissões da AWS para o AWS marketplace:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Tipo de AMI: Amazon Linux 2 (AL2_x86_64) ou Amazon Linux 2 Arm(AL2_ARM_64)
- Tipo de nó: AMD ou ARM
- Um sistema de arquivos Amazon FSx for NetApp ONTAP existente

Habilite o complemento Trident para AWS

Console de gerenciamento

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, selecione **Clusters**.
3. Selecione o nome do cluster para o qual deseja configurar o complemento Trident CSI da NetApp.
4. Selecione **Complementos** e depois selecione **Obter mais complementos**.
5. Siga estes passos para selecionar o software complementar:
   a. Desça a página até a seção **Complementos do AWS Marketplace** e digite **"Trident"** na caixa de pesquisa.
   b. Selecione a caixa de seleção no canto superior direito da caixa Trident by NetApp.
   c. Selecione **Next**.
6. Na página de configurações **Configurar add-ons selecionados**, faça o seguinte:

   OBSERVAÇÃO **Ignore estas etapas se estiver usando a associação de Pod Identity.**

   a. Selecione a **Version** que deseja usar.
   b. Se você estiver usando autenticação IRSA, certifique-se de definir os valores de configuração disponíveis nas configurações opcionais:
      - Selecione a **Version** que deseja usar.
      - Siga o **esquema de configuração do complemento** e defina o parâmetro **configurationValues** na seção **Valores de configuração** para o role-arn que você criou na etapa anterior (o valor deve estar no seguinte formato):

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

   +
   Se você selecionar Override como método de resolução de conflitos, uma ou mais configurações do add-on existente poderão ser substituídas pelas configurações do add-on do Amazon EKS. Se você não habilitar esta opção e houver um conflito com suas configurações existentes, a operação falhará. Você pode usar a mensagem de erro resultante para solucionar o conflito. Antes de selecionar esta opção, certifique-se de que o add-on do Amazon EKS não gerencie configurações que você precise gerenciar manualmente.

7. Escolha **Próximo**.
8. Na página **Revisar e adicionar**, escolha **Criar**.

   Após a conclusão da instalação do complemento, você verá o complemento instalado.

AWS CLI

1. Crie o `add-on.json` arquivo:

Para Pod Identity, utilize o seguinte formato:

OBSERVAÇÃO Use o

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

Para autenticação IRSA, utilize o seguinte formato:

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

OBSERVAÇÃO Substitua `<role ARN>` pelo ARN da função que foi criada na etapa anterior.

2. Instale o complemento Trident EKS.

```
aws eks create-addon --cli-input-json file://add-on.json
```

eksctl

O seguinte comando de exemplo instala o Trident EKS add-on:

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

Atualize o complemento Trident EKS

Console de gerenciamento

1. Abra o console do Amazon EKS https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, selecione **Clusters**.
3. Selecione o nome do cluster para o qual deseja atualizar o software complementar NetApp Trident CSI.
4. Selecione a guia **Add-ons**.
5. Selecione **Trident por NetApp** e depois selecione **Editar**.
6. Na página **Configurar Trident por NetApp**, faça o seguinte:
   a. Selecione a **Version** que deseja usar.
   b. Expanda as **Configurações opcionais de configuração** e modifique conforme necessário.
   c. Selecione **Save changes**.

AWS CLI

O exemplo a seguir atualiza o add-on EKS:

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

eksctl

- Verifique a versão atual do seu FSxN Trident CSI add-on. Substitua `my-cluster` pelo nome do seu cluster.

```
eksctl get addon --name netapp_trident-operator --cluster my-cluster
```

Exemplo de saída:

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v25.6.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- Atualize o software complementar para a versão retornada em UPDATE AVAILABLE na saída da etapa anterior.

```
eksctl update addon --name netapp_trident-operator --version
v25.6.0-eksbuild.1 --cluster my-cluster --force
```

Se você remover a `--force` opção e alguma das configurações do add-on do Amazon EKS entrar em conflito com suas configurações existentes, a atualização do add-on do Amazon EKS falhará; você receberá uma mensagem de erro para ajudá-lo a resolver o conflito. Antes de especificar esta opção, certifique-se de que o add-on do Amazon EKS não gerencie configurações que você precisa gerenciar, pois essas configurações serão sobrescritas com esta opção. Para mais informações sobre outras opções para esta configuração, consulte "Complementos". Para mais informações sobre o gerenciamento de campos do Kubernetes no Amazon EKS, consulte "Gerenciamento de campos do Kubernetes".

Desinstalar/remover o Trident EKS add-on

Você tem duas opções para remover um add-on do Amazon EKS:

- **Preservar software complementar no seu cluster** – Esta opção remove o gerenciamento de quaisquer configurações pelo Amazon EKS. Ela também remove a capacidade do Amazon EKS de notificá-lo sobre atualizações e atualizar automaticamente o add-on do Amazon EKS após você iniciar uma atualização. No entanto, ela preserva o software complementar no seu cluster. Esta opção transforma o add-on em uma instalação autogerenciada, em vez de um add-on do Amazon EKS. Com esta opção, não há tempo de inatividade para o add-on. Mantenha a `--preserve` opção no comando para preservar o add-on.
- **Remova o software complementar completamente do seu cluster** – NetApp recomenda que você remova o complemento do Amazon EKS do seu cluster somente se não houver recursos no seu cluster que dependam dele. Remova a `--preserve` opção do comando `delete` para remover o software complementar.

OBSERVAÇÃO	Se o software complementar tiver uma conta IAM associada a ele, a conta IAM não será removida.

Console de gerenciamento

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, selecione **Clusters**.
3. Selecione o nome do cluster do qual você deseja remover o complemento NetApp Trident CSI.
4. Selecione a aba **Complementos** e depois selecione **Trident by NetApp**.*
5. Selecione **Remove**.
6. Na caixa de diálogo **Remover netapp_trident-operator confirmation**, faça o seguinte:
   a. Se você deseja que o Amazon EKS pare de gerenciar as configurações do software complementar, selecione **Preservar no cluster**. Faça isso se quiser manter o software complementar no seu cluster para que você possa gerenciar todas as configurações do software complementar por conta própria.
   b. Digite **netapp_trident-operator**.
   c. Selecione **Remove**.

AWS CLI

Substitua `my-cluster` pelo nome do seu cluster e, em seguida, execute o seguinte comando.

```
aws eks delete-addon --cluster-name my-cluster --addon-name
netapp_trident-operator --preserve
```

eksctl

O comando a seguir desinstala o Trident EKS add-on:

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

Configure uma classe de armazenamento

O "Objeto Kubernetes StorageClass" identifica um provisionador e instrui o provisionador sobre como provisionar volumes. Esta seção mostra como configurar um objeto StorageClass do Kubernetes que especifica Trident como provisionador.

Criar um StorageClass Objeto

Ao criar um StorageClass para FSx for ONTAP, o Trident criará automaticamente a configuração de backend.

OBSERVAÇÃO	Se você deseja configurar manualmente o backend de armazenamento, consulte a [create-a-kubernetes-storageclass-without-automatic-backend-configuration] seção sobre como criar o backend Trident e a classe de armazenamento separadamente.

Especifique os parâmetros necessários de StorageClass

Os três parâmetros a seguir precisam ser definidos ao criar um StorageClass:

Parâmetro	Obrigatório	Tipo	Descrição
`fsxFilesystemID`	Sim	string	FSx para NetApp ONTAP ID do sistema de arquivos
`storageDriverName`	Sim	string	Driver de storage Trident (por exemplo, `ontap-nas` ou `ontap-san`)
`credentialsName`	Sim	string	Nome do segredo do Kubernetes que contém as credenciais do FSx for ONTAP

Especifique parâmetros opcionais

Você pode passar parâmetros opcionais para o backend através do StorageClass. Defina todos os valores opcionais como strings na StorageClass `parameters` section. Para obter uma lista completa dos parâmetros de backend, consulte: "Configuração do backend FSx para NetApp ONTAP".

Exemplo de arquivos de configuração StorageClass.

O exemplo a seguir mostra um StorageClass que aciona a configuração automática do backend.

YAML

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-fsx-demo
  annotations:
    description: "Demo StorageClass for FSx for NetApp ONTAP"
provisioner: csi.trident.netapp.io
parameters:
  fsxFilesystemID: "fs-0abc123"
  storageDriverName: "ontap-nas"
  credentialsName: trident-fsx-credentials
allowVolumeExpansion: true
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

JSON

```json
{
  "apiVersion": "storage.k8s.io/v1",
  "kind": "StorageClass",
  "metadata": {
    "name": "ontap-fsx-demo",
    "annotations": {
      "description": "Demo StorageClass for FSx for NetApp ONTAP"
    }
  },
  "provisioner": "csi.trident.netapp.io",
  "parameters": {
    "fsxFilesystemID": "fs-0abc123",
    "storageDriverName": "ontap-nas",
    "credentialsName": "trident-fsx-credentials"
  },
  "allowVolumeExpansion": true,
  "reclaimPolicy": "Delete",
  "volumeBindingMode": "Immediate"
}
```

Crie o StorageClass

Após criar o arquivo de configuração, execute o seguinte comando para criar a storage class.

```
kubectl create -f storage-class-ontapnas.yaml
```

Agora você deverá ver uma classe de storage **basic-csi** tanto no Kubernetes quanto no Trident, e o Trident deverá ter descoberto os pools no backend.

```
kubectl get sc basic-csi
```

```
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h
```

Após aplicar o StorageClass, Trident cria o backend automaticamente. Você pode então criar PersistentVolumeClaims que referenciam esse StorageClass.

Verifique o status da configuração do backend

Trident registra o resultado da criação do backend em anotações de StorageClass.

Anotação	Descrição
`trident.netapp.io/configuratorStatus`	Resultado da configuração (`Success` ou `Failure`)
`trident.netapp.io/configuratorMessage`	Mensagem detalhada de status ou erro
`trident.netapp.io/configuratorName`	Nome do recurso interno do configurador
`trident.netapp.io/managed`	Indica que o StorageClass é gerenciado pela Trident
`trident.netapp.io/additionalStoragePools`	Pools de storage criados para este backend

Para verificar o status, execute:

```
kubectl get storageclass ontap-fsx-demo -o yaml
```

Confirme que `trident.netapp.io/configuratorStatus` está definido como `Success`. Se o valor for `Failure`, revise `trident.netapp.io/configuratorMessage` para o erro.

Adicionar sistemas de arquivos FSxN adicionais

Se você precisar de capacidade de storage adicional enquanto continua usando o mesmo StorageClass, adicione IDs de sistema de arquivos FSxN adicionais.

Edite o StorageClass e adicione a seguinte anotação:

```
metadata:
  annotations:
    trident.netapp.io/additionalFsxNFileSystemID: '["fs-
xxxxxxxxxxxxxxxxx"]'
```

Após aplicar a alteração, Trident atualiza a configuração do backend e atualiza as anotações de StorageClass.

Considerações operacionais e limitações

- Excluir um StorageClass que possui a configuração automática de backend geralmente exclui o Trident backend associado. Isso pode interromper a conectividade de storage e interromper cargas de trabalho em execução. Valide o impacto antes de excluir um StorageClass gerenciado.
- A configuração automática de backend é suportada apenas para AWS FSx para NetApp ONTAP.

Crie um Kubernetes StorageClass sem configuração automática de backend

Se você deseja criar o backend do Trident e o StorageClass separadamente, siga estas etapas.

Entenda como funciona a configuração automática do backend

Trident deriva a configuração do backend da definição de StorageClass. Quando você aplica o StorageClass, Trident valida os parâmetros necessários, cria o backend e anota o StorageClass com o status.

Trident cria o VolumeSnapshotClass apenas uma vez. Trident reutiliza o mesmo VolumeSnapshotClass para StorageClasses subsequentes.

Crie o backend do Trident

Para criar um backend Trident, você precisa criar um arquivo de configuração em formato JSON ou YAML. O arquivo deve especificar o tipo de storage desejado (NAS ou SAN), o sistema de arquivos, a SVM de onde obtê-lo e como autenticar com ela. O exemplo a seguir mostra como definir um storage baseado em NAS e usar um segredo da AWS para armazenar as credenciais da SVM que você deseja usar:

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
    "namespace": "trident"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFilesystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

Detalhes do driver FSx for ONTAP

Você pode integrar Trident com Amazon FSx for NetApp ONTAP usando os seguintes drivers:

Nome do driver	Descrição
`ontap-san`	Cada PV provisionado é um LUN dentro de seu próprio volume Amazon FSx for NetApp ONTAP. Recomendado para armazenamento em bloco.
`ontap-nas`	Cada PV provisionado é um volume completo do Amazon FSx for NetApp ONTAP. Recomendado para NFS e SMB.
`ontap-san-economy`	Cada PV provisionado é um LUN com um número configurável de LUNs por Amazon FSx for NetApp ONTAP volume.
`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.

Para obter detalhes sobre o driver, consulte "Drivers NAS" e "Drivers SAN".

Criar o backend

Após criar o arquivo de configuração, execute os seguintes comandos para criar e validar a Trident Backend Configuration (TBC):

- Crie a configuração de backend do Trident (TBC) a partir do arquivo yaml e execute o seguinte comando:

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- Valide se a configuração do backend Trident (TBC) foi criada com sucesso:

```
Kubectl get tbc -n trident
```

```
NAME                          BACKEND NAME          BACKEND UUID
PHASE   STATUS

backend-tbc-ontap-nas         tbc-ontap-nas         933e0071-66ce-4324-
b9ff-f96d916ac5e9   Bound   Success
```

Para obter mais informações sobre outras opções de configuração, consulte a [Backend-advanced-configuration-and-examples] seção abaixo.

Configurar uma Storage Class sem configuração automática de backend

A seguir, apresentamos exemplos de configurações de Storage Class para uso com Trident e FSx for ONTAP.

Classe de armazenamento para NFS

Você pode usar este exemplo para configurar StorageClass para volumes usando NFS (consulte a seção de Atributos do Trident abaixo para obter a lista completa de atributos):

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true"
```

Classe de armazenamento para iSCSI

Use este exemplo para configurar StorageClass para volumes usando iSCSI:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  provisioningType: "thin"
  snapshots: "true"
```

Classe de armazenamento usando NFSv3 e AWS Bottlerocket

Para provisionar volumes NFSv3 no AWS Bottlerocket, adicione o `mountOptions`necessário à classe de armazenamento:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
mountOptions:
  - nfsvers=3
  - nolock
```

Atributos do Trident StorageClass

Esses parâmetros determinam quais pools de storage gerenciados pelo Trident devem ser utilizados para provisionar volumes de um determinado tipo.

Atributo	Tipo	Valores	Oferta	Solicitação	Apoiado por
mídia[1]	string	hdd, híbrido, ssd	O pool contém mídias deste tipo; híbrido significa ambos	Tipo de mídia especificado	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
provisioningType	string	fino, grosso	Pool suporta este método de provisionamento	Método de provisionamento especificado	espesso: all ontap; fino: all ontap & solidfire-san
backendType	string	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	Pool pertence a este tipo de backend	Backend especificado	Todos os drivers
instantâneos	bool	true, false	O pool suporta volumes com snapshots	Volume com snapshots ativados	ontap-nas, ontap-san, solidfire-san
clones	bool	true, false	Pool suporta clonagem de volumes	Volume com clonagem ativada	ontap-nas, ontap-san, solidfire-san

Atributo	Tipo	Valores	Oferta	Solicitação	Apoiado por
criptografia	bool	true, false	Pool suporta volumes criptografados	Volume com criptografia ativada	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	inteiro	inteiro positivo	O pool é capaz de garantir IOPS nessa faixa	Volume garantiu esses IOPS	solidfire-san

[1]: Não suportado pelo ONTAP Select ou FSx for ONTAP systems

Consulte "Objetos Kubernetes e Trident" para obter detalhes sobre como as classes de armazenamento interagem com o `PersistentVolumeClaim` e os parâmetros para controlar como Trident provisiona volumes.

Crie a classe de armazenamento

Depois de configurar o StorageClass, você pode criá-lo no Kubernetes.

Passos

1. Este é um objeto do Kubernetes, portanto, use `kubectl` para criá-lo no Kubernetes.

```
kubectl create -f storage-class-ontapnas.yaml
```

2. Agora você deverá ver uma classe de storage **basic-csi** tanto no Kubernetes quanto no Trident, e o Trident deverá ter descoberto os pools no backend.

```
kubectl get sc basic-csi
```

```
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h
```

Provisionar volumes SMB

Você pode provisionar volumes SMB usando o `ontap-nas` driver. No entanto, para isso, você deve concluir estas etapas: "Prepare-se para provisionar volumes SMB".

Configuração avançada do backend e exemplos

Consulte a tabela a seguir para as opções de configuração do backend:

Parâmetro	Descrição	Exemplo
`version`		Sempre 1

Parâmetro	Descrição	Exemplo
`storageDriverName`	Nome do driver de armazenamento	`ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`
`backendName`	Nome personalizado ou o storage backend	Nome do driver + "_" + dataLIF
`managementLIF`	Endereço IP de um cluster ou LIF de gerenciamento de SVM. Um nome de domínio totalmente qualificado (FQDN) pode ser especificado. Pode ser configurado para usar endereços IPv6 se Trident foi instalado com o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Se você fornecer o `fsxFilesystemID` sob o campo `aws`, não é necessário fornecer o `managementLIF`, pois Trident recupera as informações da SVM `managementLIF` da AWS. Portanto, você deve fornecer credenciais para um usuário na SVM (por exemplo: vsadmin) e o usuário deve ter a função `vsadmin`.	"10.0.0.1", "[2001:1234:abcd::fefe]"

Parâmetro	Descrição	Exemplo
`dataLIF`	Endereço IP do protocolo LIF. **ONTAP NAS drivers**: NetApp recomenda especificar dataLIF. Caso não seja fornecido, Trident busca os dataLIFs da SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) para ser usado nas operações de montagem NFS, permitindo criar um DNS round-robin para balancear a carga entre vários dataLIFs. Pode ser alterado após a configuração inicial. **ONTAP SAN drivers**: não especifique para iSCSI. Trident usa ONTAP Selective LUN Map para descobrir os LIFs iSCSI necessários para estabelecer uma sessão multipath. Um aviso é gerado se dataLIF for definido explicitamente. Pode ser configurado para usar endereços IPv6 se Trident foi instalado com o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	
`autoExportPolicy`	Ativar a criação e atualização automática de políticas de exportação [Booleano]. Usando as opções `autoExportPolicy` e `autoExportCIDRs`, Trident pode gerenciar políticas de exportação automaticamente.	`false`
`autoExportCIDRs`	Lista de CIDRs para filtrar os IPs dos nós do Kubernetes quando `autoExportPolicy` estiver habilitado. Usando as opções `autoExportPolicy` e `autoExportCIDRs`, Trident pode gerenciar políticas de exportação automaticamente.	"["0.0.0.0/0", "::/0"]"
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar aos volumes	""
`clientCertificate`	Valor codificado em Base64 do certificado do cliente. Usado para autenticação baseada em certificado	""

Parâmetro	Descrição	Exemplo
`clientPrivateKey`	Valor codificado em Base64 da chave privada do cliente. Usado para autenticação baseada em certificado	""
`trustedCACertificate`	Valor codificado em Base64 do certificado da CA confiável. Opcional. Usado para autenticação baseada em certificado.	""
`username`	Nome de usuário para conectar-se ao cluster ou SVM. Usada para autenticação baseada em credenciais. Por exemplo, vsadmin.	
`password`	Senha para conectar-se ao cluster ou SVM. Usada para autenticação baseada em credenciais.	
`svm`	Máquina virtual de storage para usar	Derivado se um SVM managementLIF for especificado.
`storagePrefix`	Prefixo usado ao provisionar novos volumes no SVM. Não pode ser modificado após a criação. Para atualizar este parâmetro, você precisará criar um novo backend.	`trident`
`limitAggregateUsage`	**Não especifique para Amazon FSx para NetApp ONTAP.** As configurações fornecidas `fsxadmin` e `vsadmin` não contêm as permissões necessárias para recuperar o uso agregado e limitá-lo usando Trident.	Não use.
`limitVolumeSize`	O provisionamento falha se o tamanho do volume solicitado for superior a este valor. Também restringe o tamanho máximo dos volumes que gerencia para qtrees e LUNs, e a `qtreesPerFlexvol` opção permite personalizar o número máximo de qtrees por FlexVol volume	"" (não aplicado por padrão)
`lunsPerFlexvol`	LUNs máximas por FlexVol volume, deve estar no intervalo [50, 200]. Somente SAN.	“100”

Parâmetro	Descrição	Exemplo
`debugTraceFlags`	Sinalizadores de depuração para usar na resolução de problemas. Exemplo, {"api":false, "method":true} não use `debugTraceFlags` a menos que esteja solucionando problemas e precise de um despejo de log detalhado.	null
`nfsMountOptions`	Lista separada por vírgulas de opções de montagem NFS. As opções de montagem para volumes persistentes do Kubernetes são normalmente especificadas nas classes de armazenamento, mas se nenhuma opção de montagem for especificada em uma classe de armazenamento, Trident usará as opções de montagem especificadas no arquivo de configuração do backend de storage. Se nenhuma opção de montagem for especificada na classe de armazenamento ou no arquivo de configuração, Trident não definirá nenhuma opção de montagem em um volume persistente associado.	""
`nasType`	Configurar a criação de volumes NFS ou SMB. As opções são `nfs`, `smb` ou null. **Deve ser definido como `smb` para volumes SMB.** Definir como null define volumes NFS por padrão.	`nfs`
`qtreesPerFlexvol`	Número máximo de qtrees por FlexVol volume, deve estar no intervalo [50, 300]	`"200"`
`smbShare`	Você pode especificar uma das seguintes opções: o nome de um compartilhamento SMB criado usando o Microsoft Management Console ou ONTAP CLI, ou um nome para permitir que Trident crie o compartilhamento SMB. Este parâmetro é obrigatório para Amazon FSx for ONTAP backends.	`smb-share`

Parâmetro	Descrição	Exemplo
`useREST`	Parâmetro booleano para usar as ONTAP REST APIs. Quando definido como `true`, Trident usará as ONTAP REST APIs para se comunicar com o backend. Este recurso requer ONTAP 9.11.1 e versões posteriores. Além disso, a função de login do ONTAP utilizada deve ter acesso ao aplicativo `ontap`. Isso é atendido pelas funções predefinidas `vsadmin` e `cluster-admin`.	`false`
`aws`	Você pode especificar o seguinte no arquivo de configuração do AWS FSx para ONTAP: - `fsxFilesystemID`: especificar o ID do sistema de arquivos AWS FSx. - `apiRegion`: nome da região da API da AWS. - `apikey`: chave da API da AWS. - `secretKey`: chave secreta da AWS.	`""` `""` `""`
`credentials`	Especifique as credenciais do FSx SVM a serem armazenadas no AWS Secrets Manager. - `name`: Amazon Resource Name (ARN) do segredo, que contém as credenciais do SVM. - `type`: Defina como `awsarn`. Consulte "Crie um segredo do AWS Secrets Manager" para mais informações.	

Opções de configuração de backend para provisionamento de volumes

Você pode controlar o provisionamento padrão usando essas opções na seção `defaults` do arquivo de configuração. Para um exemplo, veja os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
`spaceAllocation`	Alocação de espaço para LUNs	`true`
`spaceReserve`	Modo de reserva de espaço; "none" (fino) ou "volume" (grosso)	`none`
`snapshotPolicy`	Política do Snapshot a ser usada	`none`

Parâmetro	Descrição	Padrão
`qosPolicy`	Grupo de políticas de QoS a ser atribuído aos volumes criados. Escolha uma das opções qosPolicy ou adaptiveQosPolicy por pool de storage ou backend. O uso de grupos de políticas de QoS com Trident requer ONTAP 9.8 ou posterior. Você deve usar um grupo de políticas de QoS não compartilhado e garantir que o grupo de políticas seja aplicado a cada componente individualmente. Um grupo de políticas de QoS compartilhado impõe o limite máximo para a taxa de transferência total de todas as cargas de trabalho.	""
`adaptiveQosPolicy`	Grupo de políticas de QoS adaptável para atribuir aos volumes criados. Escolha uma das opções qosPolicy ou adaptiveQosPolicy por pool de storage ou backend. Não compatível com ontap-nas-economy.	""
`snapshotReserve`	Porcentagem do volume reservada para snapshots "0"	Se `snapshotPolicy` for `none`, `else` ""
`splitOnClone`	Separar um clone de seu progenitor no momento da criação	`false`
`encryption`	Habilite NetApp Volume Encryption (NVE) no novo volume; o padrão é `false`. A NVE deve estar licenciada e habilitada no cluster para usar esta opção. Se a NAE estiver habilitada no backend, qualquer volume provisionado no Trident terá a NAE habilitada. Para mais informações, consulte: "Como Trident funciona com NVE e NAE".	`false`
`luksEncryption`	Ative a criptografia LUKS. Consulte "Use Linux Unified Key Setup (LUKS)". Somente SAN.	""
`tieringPolicy`	Política de tiering a ser usada `none`	
`unixPermissions`	Modo para novos volumes. **Deixe em branco para volumes SMB.**	""

Parâmetro	Descrição	Padrão
`securityStyle`	Estilo de segurança para novos volumes. NFS suporta `mixed` e `unix` estilos de segurança. SMB suporta `mixed` e `ntfs` estilos de segurança.	NFS default é `unix`. SMB default é `ntfs`.

Configurar um PVC

Esta seção inclui instruções sobre como criar um PersistentVolumeClaim (PVC) que usa o StorageClass do Kubernetes configurado para solicitar um PV. Se a solicitação for bem-sucedida, você poderá montar o PV em um pod.

Crie o PVC

Um "*PersistentVolumeClaim*" (PVC) é uma solicitação de acesso ao PersistentVolume no cluster. O PVC pode ser configurado para solicitar armazenamento de um determinado tamanho ou modo de acesso. Usando o StorageClass associado, o administrador do cluster pode controlar mais do que apenas o tamanho e o modo de acesso do PersistentVolume—como desempenho ou nível de serviço.

Após criar o backend Trident e o StorageClass, você pode criar um PVC. Depois que o PVC for criado, você pode montar o volume em um pod.

Exemplos de manifestos

Os exemplos a seguir mostram opções básicas de configuração de PVC.

PVC com acesso RWX

Este exemplo mostra um PVC básico com acesso RWX associado a um StorageClass chamado `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-gold
```

Exemplo de PVC usando iSCSI

Este exemplo mostra um PVC básico para iSCSI com acesso RWO que está associado a um StorageClass chamado `protection-gold`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: protection-gold
```

Criar PVC

Passos

1. Crie o PVC.

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

2. Verifique o status do PVC.

```
kubectl get pvc
```

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

Consulte "Objetos Kubernetes e Trident" para obter detalhes sobre como as classes de armazenamento interagem com o `PersistentVolumeClaim` e os parâmetros para controlar como Trident provisiona volumes.

Implantar um aplicativo

Após a criação da classe de armazenamento e do PVC, você pode montar o PV em um pod. Esta seção lista o comando de exemplo e a configuração para anexar o PV a um pod.

Implante um aplicativo de exemplo

Passos

1. Monte o volume em um pod.

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

Estes exemplos mostram configurações básicas para conectar o PVC a um pod: **configuração básica**:

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

OBSERVAÇÃO Você pode monitorar o progresso usando `kubectl get pod --watch`.

2. Verifique se o volume está montado em `/my/mount/path`.

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

```
Filesystem                                                      Size
Used  Avail Use%  Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06  1.1G
320K  1.0G   1%   /my/mount/path
```

Agora você pode excluir o Pod. O aplicativo Pod deixará de existir, mas o volume permanecerá.

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

Configurar o complemento Trident EKS em um cluster EKS

NetApp Trident simplifica o gerenciamento de storage do Amazon FSx for NetApp ONTAP no Kubernetes para permitir que seus desenvolvedores e administradores se concentrem na implantação de aplicativos. O NetApp Trident EKS add-on inclui os

patches de segurança mais recentes, correções de bugs e é validado pela AWS para funcionar com o Amazon EKS. O EKS add-on permite garantir de forma consistente que seus clusters Amazon EKS estejam seguros e estáveis e reduz a quantidade de trabalho necessária para instalar, configurar e atualizar add-ons.

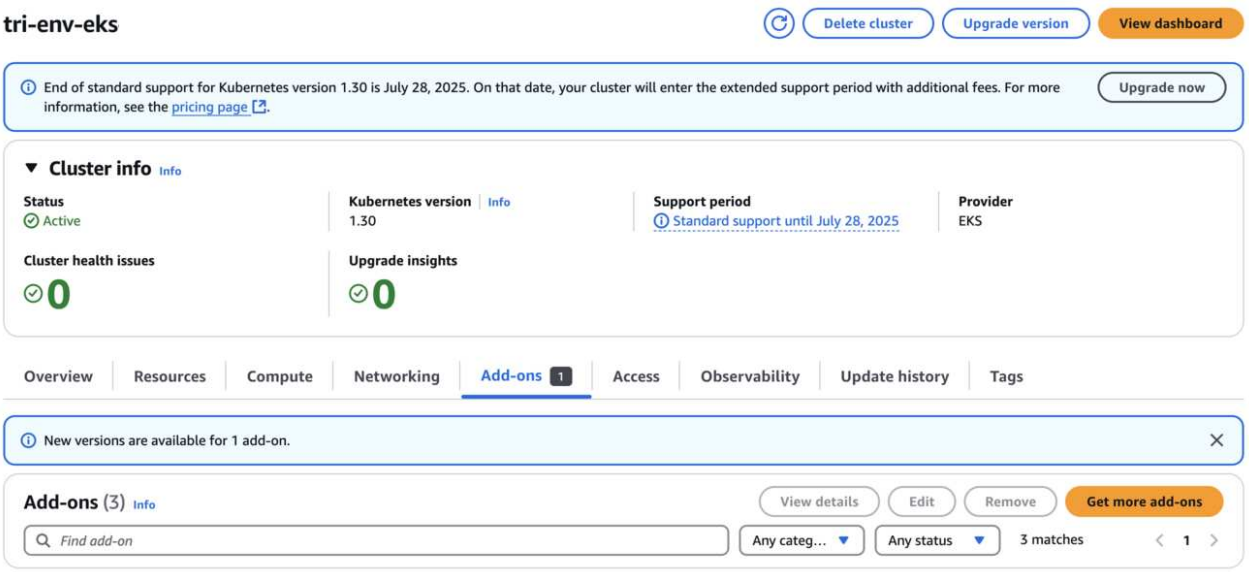
Pré-requisitos

Certifique-se de ter o seguinte antes de configurar o add-on Trident para AWS EKS:

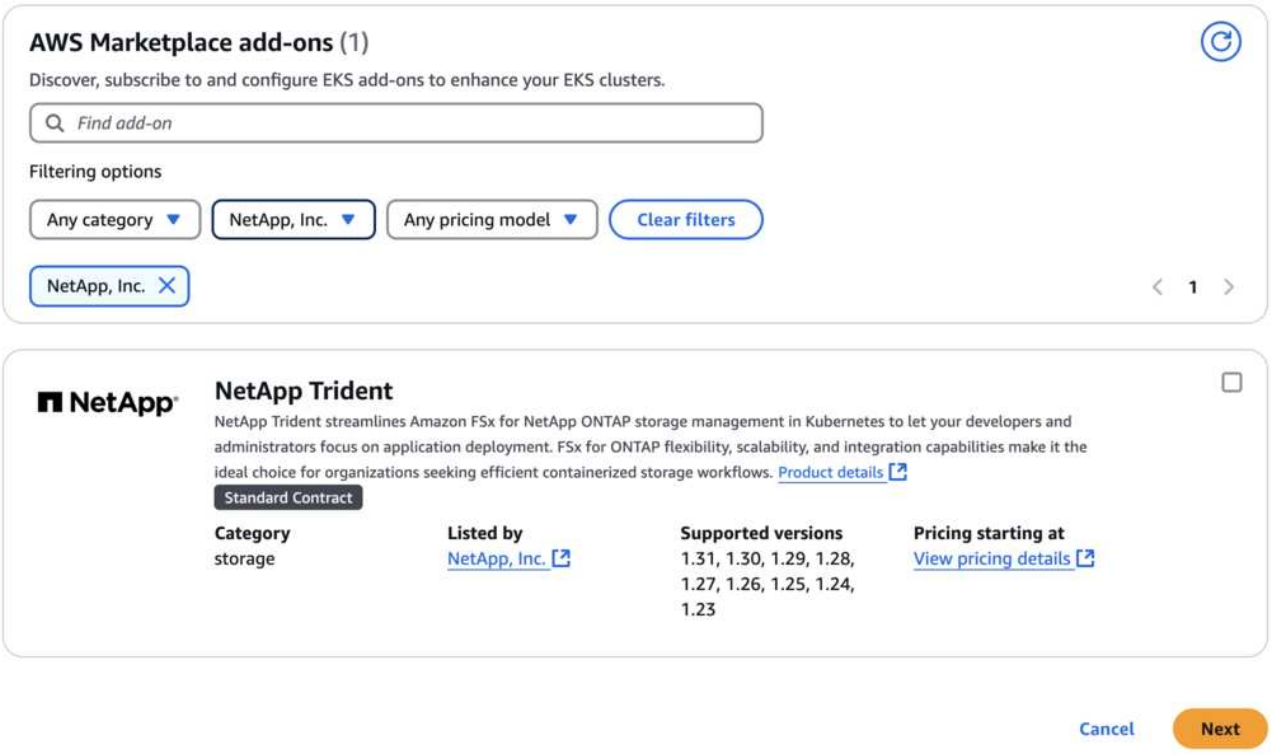
- Uma conta de cluster Amazon EKS com permissões para trabalhar com complementos. Consulte "Complementos do Amazon EKS".
- Permissões da AWS para o AWS marketplace:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Tipo de AMI: Amazon Linux 2 (AL2_x86_64) ou Amazon Linux 2 Arm(AL2_ARM_64)
- Tipo de nó: AMD ou ARM
- Um sistema de arquivos Amazon FSx for NetApp ONTAP existente

Passos

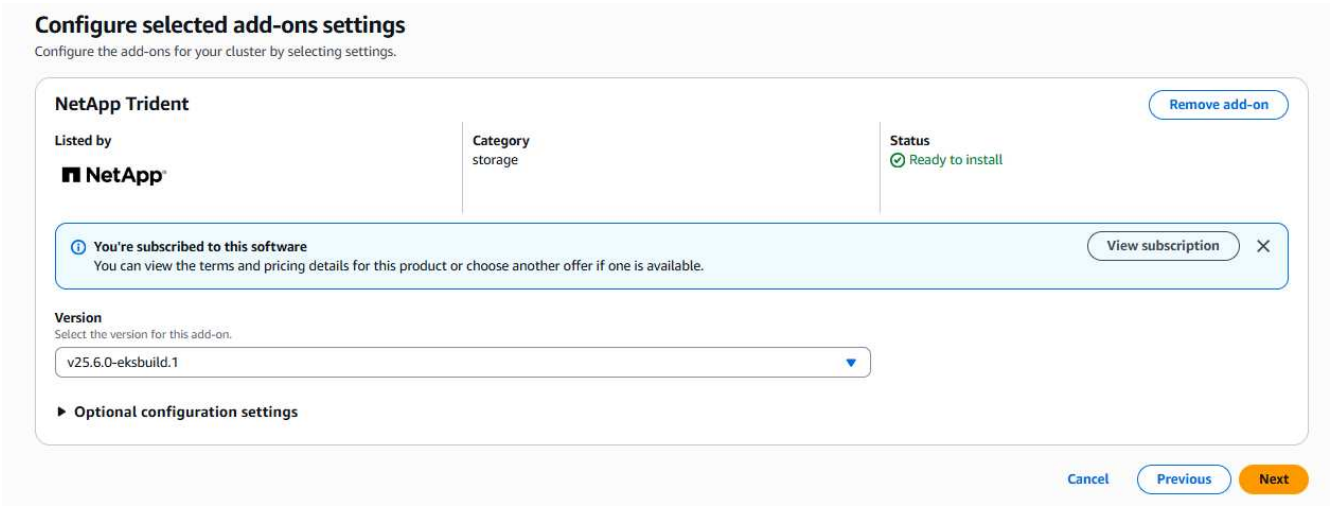
1. Certifique-se de criar uma função do IAM e um segredo da AWS para permitir que os pods do EKS acessem os recursos da AWS. Para obter instruções, consulte "Crie uma função do IAM e um segredo da AWS".
2. No seu cluster Kubernetes EKS, navegue até a guia **Add-ons**.



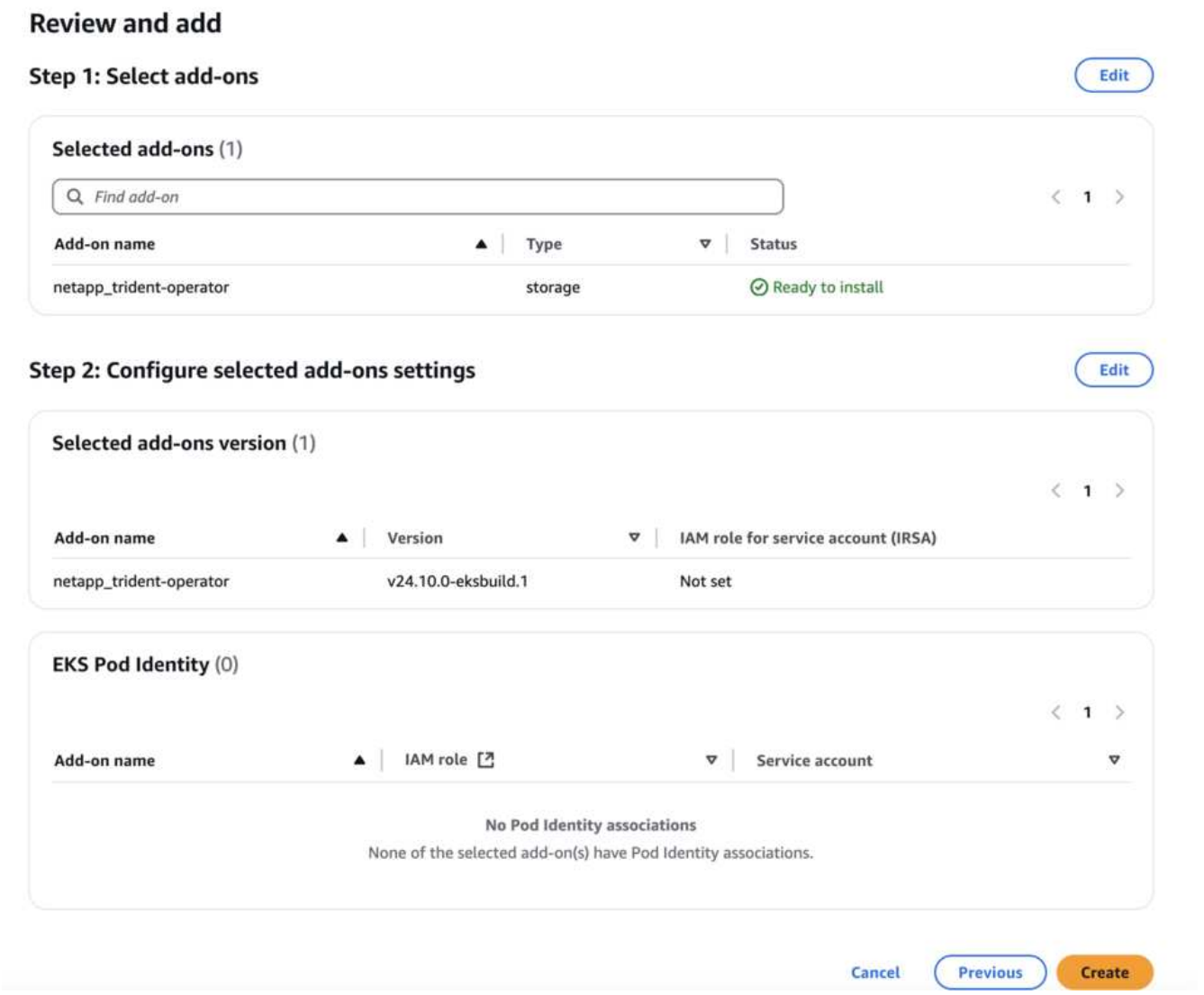
3. Acesse **AWS Marketplace add-ons** e escolha a categoria *storage*.



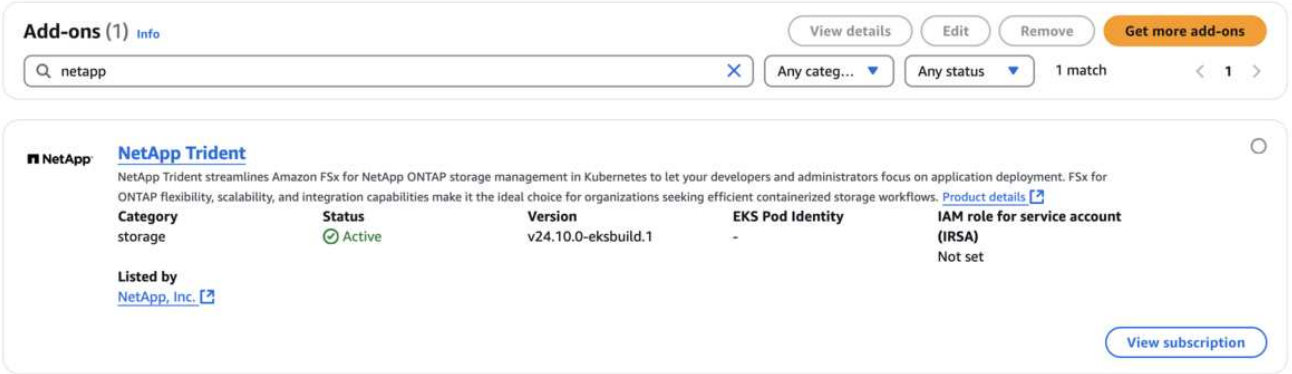
4. Localize **NetApp Trident** e selecione a caixa de seleção para o add-on Trident e clique em **Próximo**.
5. Escolha a versão desejada do add-on.



6. Configurar as definições adicionais necessárias.



7. Se você estiver usando IRSA (IAM roles para conta de serviço), consulte as etapas de configuração adicionais "aqui".
8. Selecione **Create**.
9. Verifique se o status do add-on é *Ativo*.



10. Execute o seguinte comando para verificar se Trident está instalado corretamente no cluster:

```
kubectl get pods -n trident
```

11. Continue a configuração e configure o backend de storage. Para obter informações, consulte "Configurar o backend de armazenamento".

Instalar/desinstalar o Trident EKS add-on usando CLI

Instale o NetApp Trident EKS add-on usando CLI:

O seguinte comando de exemplo instala o Trident EKS add-on:
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.0-eksbuild.1` (com uma versão dedicada)

O seguinte comando de exemplo instala o Trident EKS add-on versão 25.6.1:
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.1-eksbuild.1` (com uma versão dedicada)

O seguinte comando de exemplo instala o Trident EKS add-on versão 25.6.2:
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.2-eksbuild.1` (com uma versão dedicada)

Desinstale o complemento NetApp Trident EKS usando a CLI:

O comando a seguir desinstala o Trident EKS add-on:

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

Criar backends com kubectl

Um backend define a relação entre Trident e um sistema de storage. Ele informa ao Trident como se comunicar com esse sistema de storage e como o Trident deve provisionar volumes a partir dele. Após a instalação do Trident, o próximo passo é criar um backend. A `TridentBackendConfig` Custom Resource Definition (CRD) permite que você crie e gerencie backends do Trident diretamente pela interface do Kubernetes. Você pode fazer isso usando `kubectl` ou a ferramenta de linha de comando equivalente para sua distribuição do Kubernetes.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`) é um CRD de frontend com namespace que permite gerenciar backends do Trident usando `kubectl`. Administradores de Kubernetes e de storage agora podem criar e gerenciar backends diretamente por meio da CLI do Kubernetes sem a necessidade de um utilitário de linha de comando dedicado (`tridentctl`).

Ao criar um `TridentBackendConfig` objeto, ocorre o seguinte:

- Um backend é criado automaticamente pelo Trident com base na configuração que você fornece. Isso é representado internamente como um `TridentBackend` (`tbe`, `tridentbackend`) CR.
- O `TridentBackendConfig` está exclusivamente ligado a um `TridentBackend` que foi criado pela

Trident.

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

AVISO	`TridentBackend` Os CRs são criados automaticamente pelo Trident. Você **não deve** modificá-los. Se você quiser fazer atualizações nos 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 consultar os exemplos no "trident-installer" diretório para obter configurações de amostra para a plataforma de storage/serviço desejado.

O `spec` aceita parâmetros de configuração específicos do backend. Neste exemplo, o backend usa o `ontap-san` driver de storage e utiliza os parâmetros de configuração que estão tabulados aqui. Para a lista de opções de configuração para o driver de storage desejado, consulte o "Informações de configuração do backend para o seu driver de storage".

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

- `credentials`: Este parâmetro é obrigatório e contém as credenciais usadas para autenticação com o sistema de storage/serviço. 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` for excluído. Ele pode assumir um dos dois valores possíveis:
  - `delete`: Isso resulta na exclusão tanto do `TridentBackendConfig`CR quanto do backend associado. Este é o valor padrão.
  - `retain`: Quando uma `TridentBackendConfig` CR é excluída, a definição do backend ainda estará presente e poderá ser gerenciada com `tridentctl`. Definir a política de exclusão para `retain` permite que os usuários façam downgrade para uma versão anterior (pré-21.04) e mantenham os backends criados. O valor deste campo pode ser atualizado após uma `TridentBackendConfig` CR

ser criada.

OBSERVAÇÃO	O nome de um backend é definido usando `spec.backendName`. Se não for especificado, o nome do backend é definido como o nome do objeto `TridentBackendConfig` (metadata.name). Recomenda-se definir explicitamente os nomes dos backends usando `spec.backendName`.
DICA	Os backends que foram criados com `tridentctl` não possuem um objeto `TridentBackendConfig` associado. Você pode optar por gerenciar esses backends com `kubectl` criando um `TridentBackendConfig` CR. É preciso ter cuidado para especificar parâmetros de configuração idênticos (como `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName` e assim por diante). O Trident vinculará automaticamente o novo `TridentBackendConfig` ao backend preexistente.

Visão geral das etapas

Para criar um novo backend usando `kubectl`, você deve fazer o seguinte:

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

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

Passo 1: criar um segredo do Kubernetes

Crie um segredo que contenha as credenciais de acesso ao backend. Isso é exclusivo para cada serviço/plataforma de storage. Veja 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: password
```

Esta tabela resume os campos que devem ser incluídos no Secret para cada plataforma de storage:

Descrição dos campos secretos da plataforma de storage	Segredo	Descrição dos campos
Azure NetApp Files	clientID	O client ID de um registro de aplicativo
Element (NetApp HCI/SolidFire)	Endpoint	MVIP para o SolidFire cluster com credenciais de locatário
ONTAP	nome de usuário	Nome de usuário para conectar-se ao cluster/SVM. Usada para autenticação baseada em credencial
ONTAP	senha	Senha para conectar-se ao cluster/SVM. Usada para autenticação baseada em credencial
ONTAP	clientPrivateKey	Valor codificado em Base64 da chave privada do cliente. Usado para autenticação baseada em certificado
ONTAP	chapUsername	Nome de usuário de entrada. Obrigatório se useCHAP=true. Para `ontap-san` e `ontap-san-economy`
ONTAP	chapInitiatorSecret	Segredo do iniciador CHAP. Obrigatório se useCHAP=true. Para `ontap-san` e `ontap-san-economy`
ONTAP	chapTargetUsername	Nome de usuário de destino. Obrigatório se useCHAP=true. Para `ontap-san` e `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	Segredo do iniciador do alvo CHAP. Obrigatório se useCHAP=true. Para `ontap-san` e `ontap-san-economy`

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

Etapa 2: Criar o `TridentBackendConfig` CR

Agora você está pronto para criar seu `TridentBackendConfig` CR. Neste exemplo, um backend que utiliza o `ontap-san` 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: Verificar o status do `TridentBackendConfig`CR

Agora que você criou o `TridentBackendConfig` CR, pode verificar o status. 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 e vinculado com sucesso ao `TridentBackendConfig` CR.

A fase pode assumir um dos seguintes valores:

- `Bound`: O `TridentBackendConfig` CR está associado a um backend, e esse backend contém `configRef` definido para o `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 a mudança de fase, ele não pode retornar ao estado Unbound novamente.
- `Deleting`: O `TridentBackendConfig` CR `deletionPolicy` foi configurado para ser excluído. Quando o `TridentBackendConfig` CR é excluído, ele transita para o estado Deleting.
  - Caso não existam reivindicações de volume persistentes (PVCs) no backend, excluir o `TridentBackendConfig` resultará na exclusão do backend pelo Trident, bem como da

`TridentBackendConfig` CR.

  - Se um ou mais PVCs estiverem presentes no backend, ele entra em estado de exclusão. O `TridentBackendConfig` CR subsequentemente também entra na fase de exclusão. O backend e `TridentBackendConfig` são excluídos somente após todos os PVCs serem excluídos.

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

Nesta etapa, um backend foi criado com sucesso! Existem diversas operações que também podem ser realizadas, como "atualizações de backend e exclusões de backend".

(Opcional) Passo 4: obtenha mais detalhes

Você pode executar o seguinte comando para obter mais informações sobre seu backend:

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

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

Além disso, você também pode obter um dump YAML/JSON de `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 backend criado em resposta ao `TridentBackendConfig` CR. O `lastOperationStatus` campo representa o status da última operação do `TridentBackendConfig` CR, que pode ser iniciada pelo usuário (por exemplo, o usuário alterou algo em `spec`) ou iniciada pelo Trident (por exemplo, durante reinicializações do Trident). Pode ser Success ou Failed. `phase` representa o status da relação entre o `TridentBackendConfig` CR e o backend. No exemplo acima, `phase` tem o valor Bound, o que significa que o `TridentBackendConfig` CR está associado ao backend.

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

AVISO Não é possível atualizar ou excluir um backend que contenha um objeto associado `TridentBackendConfig` usando `tridentctl`. Para entender as etapas envolvidas na troca entre `tridentctl` e `TridentBackendConfig`, "veja aqui".

Gerenciar backends

Realize o gerenciamento de backend com kubectl

Saiba como realizar operações de gerenciamento de backend usando `kubectl`.

Excluir um backend

Ao excluir um `TridentBackendConfig`, você instrui Trident a excluir/manter backends (com base em `deletionPolicy`). Para excluir um backend, certifique-se de que `deletionPolicy` esteja definido como delete. Para excluir apenas o `TridentBackendConfig`, certifique-se de que `deletionPolicy` esteja definido como retain. Isso garante que o backend ainda esteja presente e possa ser gerenciado usando `tridentctl`.

Execute o seguinte comando:

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

Trident não exclui os segredos do Kubernetes que estavam em uso por `TridentBackendConfig`. O usuário do Kubernetes é responsável por limpar os segredos. É preciso ter cuidado ao excluir segredos. Você só deve excluir segredos 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` para obter uma lista de todos os backends existentes. Essa lista também incluirá backends que foram criados com `tridentctl`.

Atualizar um backend

Podem haver vários motivos para atualizar um backend:

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

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

- Parâmetros (como o nome da SVM do ONTAP que está sendo usada) precisam ser atualizados.
  - Você pode atualizar `TridentBackendConfig` objetos diretamente pelo Kubernetes usando o seguinte comando:

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

- Alternativamente, você pode fazer alterações no CR existente `TridentBackendConfig` usando o seguinte comando:

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

OBSERVAÇÃO

- Se uma atualização do backend falhar, o backend continuará em sua última configuração conhecida. Você pode visualizar os logs para determinar a causa executando `kubectl get tbc <tbc-name> -o yaml -n trident` ou `kubectl describe tbc <tbc-name> -n trident`.
- Após identificar e corrigir o problema com o arquivo de configuração, você pode executar novamente o comando de atualização.

Realize o gerenciamento de backend com tridentctl

Saiba como realizar operações de gerenciamento de backend usando `tridentctl`.

Criar um backend

Após criar um "arquivo de configuração backend", 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 visualizar os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs -n trident
```

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

Excluir um backend

Para excluir um backend do Trident, faça o seguinte:

1. Recupere o nome do backend:

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

2. Exclua o backend:

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

OBSERVAÇÃO	Se Trident tiver provisionado volumes e snapshots desse backend que ainda existam, a exclusão do backend impede que novos volumes sejam provisionados por ele. O backend continuará existindo no estado "Deleting".

Veja os backends existentes

Para visualizar os backends que 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
```

Atualizar um backend

Após criar um novo arquivo de configuração backend, 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 visualizar os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs -n trident
```

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

Identifique as classes de armazenamento que usam um backend

Este é um exemplo do tipo de perguntas que você pode responder com o JSON que `tridentctl` gera para objetos do backend. Isso usa o utilitário `jq`, 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`.

Alternar entre opções de gerenciamento de backend

Aprenda sobre as diferentes maneiras de gerenciar backends no Trident.

Opções para gerenciamento de backends

Com a introdução de `TridentBackendConfig`, os administradores agora têm duas maneiras únicas de gerenciar backends. Isso levanta as seguintes questões:

- Backends criados usando `tridentctl` podem ser gerenciados com `TridentBackendConfig`?
- Backends criados usando `TridentBackendConfig` podem ser gerenciados usando `tridentctl`?

Gerencie `tridentctl` back-ends usando `TridentBackendConfig`

Esta seção aborda os passos necessários para gerenciar backends que foram criados usando `tridentctl` diretamente através da interface do Kubernetes, criando `TridentBackendConfig` objetos.

Isso se aplica aos seguintes cenários:

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

Em ambos os cenários, os backends continuarão presentes, com Trident agendando volumes e operando sobre eles. Os administradores têm uma das duas opções aqui:

- Continue usando `tridentctl` para gerenciar os backends que foram criados com ele.
- Vincule os backends criados usando `tridentctl` a um novo `TridentBackendConfig` objeto. Fazer isso significa que os backends serão gerenciados usando `kubectl` e não `tridentctl`.

Para gerenciar um backend preexistente usando `kubectl`, você precisará criar um `TridentBackendConfig` que se conecte ao backend existente. Aqui está uma visão geral de como isso funciona:

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

Etapa 0: identificar o backend

Para criar uma `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"
      }
    }
  ]
}
```

Passo 1: criar um segredo do Kubernetes

Crie um segredo que contenha as credenciais para o backend, conforme 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
```

Etapa 2: Criar um `TridentBackendConfig` CR

O próximo passo é criar uma `TridentBackendConfig` CR que se vinculará automaticamente à já existente `ontap-nas-backend` (como neste exemplo). Certifique-se de que os seguintes requisitos sejam atendidos:

- O mesmo nome de backend está definido em `spec.backendName`.
- Os parâmetros de configuração são idênticos aos do backend original.
- Pools virtuais (se presentes) devem manter a mesma ordem que no backend original.
- As credenciais são fornecidas por meio de um Kubernetes Secret e não em texto simples.

Nesse caso, o `TridentBackendConfig` ficará assim:

```
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: Verificar o status do `TridentBackendConfig`CR

Após o `TridentBackendConfig` ter sido criado, sua fase deve ser `Bound`. Ele também deve refletir o mesmo nome de backend e UUID do backend existente.

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

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

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

Gerencie `TridentBackendConfig` back-ends usando `tridentctl`

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

Etapa 0: identificar 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, observa-se que `TridentBackendConfig` foi criado com sucesso e está vinculado a um backend [observe o UUID do backend].

Etapa 1: confirmar `deletionPolicy` **está definido como** `retain`

Vamos analisar o valor de `deletionPolicy`. Isso precisa ser definido como `retain`. Isso garante que, quando um `TridentBackendConfig` CR for excluído, a definição de backend ainda estará presente e poderá ser gerenciada com `tridentctl`.

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

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

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

OBSERVAÇÃO	Não prossiga para a próxima etapa a menos que `deletionPolicy` esteja definido como `retain`.

Passo 2: Exclua o `TridentBackendConfig` CR

A etapa final é excluir o `TridentBackendConfig` CR. Após confirmar que o `deletionPolicy` está definido como `retain`, você pode prosseguir com a exclusão:

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

Ao excluir o `TridentBackendConfig` objeto, Trident simplesmente o remove sem realmente excluir o próprio backend.

Criar e gerenciar classes de armazenamento

Crie uma storage class

Configure um objeto StorageClass do Kubernetes e crie a classe de armazenamento para instruir Trident sobre como provisionar volumes.

Configure um objeto StorageClass do Kubernetes

O "Objeto Kubernetes StorageClass"identifica Trident como o provisionador usado para essa classe e instrui Trident sobre como provisionar um volume. Por exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
mountOptions:
  - nfsvers=3
  - nolock
parameters:
  backendType: "ontap-nas"
  media: "ssd"
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Consulte "Objetos Kubernetes e Trident" para obter detalhes sobre como as classes de armazenamento interagem com o `PersistentVolumeClaim` e os parâmetros para controlar como Trident provisiona volumes.

Crie uma storage class

Após criar o StorageClass objeto, você pode criar a classe de armazenamento. Exemplos de storage class fornece alguns exemplos básicos que você pode usar ou modificar.

Passos

1. Este é um objeto do Kubernetes, portanto, use `kubectl` para criá-lo no Kubernetes.

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

2. Agora você deverá ver uma classe de storage **basic-csi** tanto no Kubernetes quanto no Trident, e o Trident deverá ter descoberto os pools no backend.

```
kubectl get sc basic-csi
```

```
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h
```

```
./tridentctl -n trident get storageclass basic-csi -o json
```

```
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```


Exemplos de storage class

Trident fornece "Definições simples de storage class para backends específicos".

Alternativamente, você pode editar `sample-input/storage-class-csi.yaml.templ` o arquivo que acompanha o instalador e substituir *`BACKEND_TYPE`* pelo nome do driver de storage.

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

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

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

Gerenciar classes de armazenamento

Você pode visualizar as classes de armazenamento existentes, definir uma classe de armazenamento padrão, identificar o backend da classe de armazenamento e excluir classes de armazenamento.

Veja as classes de armazenamento existentes

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

```
kubectl get storageclass
```

- Para visualizar os detalhes da storage class do Kubernetes, execute o seguinte comando:

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

- Para visualizar as classes de armazenamento sincronizadas do Trident, execute o seguinte comando:

```
tridentctl get storageclass
```

- Para visualizar os detalhes da classe de armazenamento sincronizado do 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 armazenamento padrão. Essa é a classe de armazenamento que será usada para provisionar um Volume Persistente caso o usuário não especifique uma em uma Solicitação de Volume Persistente (PVC).

- Defina uma classe de armazenamento padrão configurando 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 a 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 da classe de armazenamento padrão usando o seguinte comando:

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

Existem também exemplos no pacote de instalação do Trident que incluem essa anotação.

OBSERVAÇÃO	Deve haver apenas uma classe de armazenamento padrão em seu cluster por vez. O Kubernetes não impede tecnicamente que você tenha mais de uma, mas se comportará como se não houvesse classe de armazenamento padrão.

Identifique o backend para uma storage class

Este é um exemplo do tipo de pergunta que você pode responder com o JSON que `tridentctl` gera para objetos de backend do Trident. Isso utiliza o utilitário `jq`, que talvez você precise instalar primeiro.

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

Excluir uma storage class

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.

Quaisquer volumes persistentes que foram criados por meio dessa classe de armazenamento permanecerão intactos, e Trident continuará a gerenciá-los.

OBSERVAÇÃO	Trident impõe um espaço em branco `fsType` para os volumes que cria. Para backends iSCSI, recomenda-se impor `parameters.fsType` no StorageClass. Você deve excluir as StorageClasses existentes e recriá-las com `parameters.fsType` especificado.

Provisionar e gerenciar volumes

Provisionar um volume

Crie um PersistentVolumeClaim (PVC) que utiliza o StorageClass do Kubernetes configurado para solicitar acesso ao PV. Em seguida, você pode montar o PV em um pod.

Visão geral

Um "*PersistentVolumeClaim*" (PVC) é uma solicitação de acesso ao PersistentVolume no cluster.

O PVC pode ser configurado para solicitar armazenamento de um determinado tamanho ou modo de acesso. Usando o StorageClass associado, o administrador do cluster pode controlar mais do que apenas o tamanho e o modo de acesso do PersistentVolume—como desempenho ou nível de serviço.

Após criar o PVC, você pode montar o volume em um pod.

Crie o PVC

Passos

1. Crie o PVC.

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

2. Verifique o status do PVC.

```
kubectl get pvc
```

```
NAME          STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS AGE
pvc-storage   Bound    pv-name    1Gi        RWO                    5m
```

1. Monte o volume em um pod.

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

OBSERVAÇÃO Você pode monitorar o progresso usando `kubectl get pod --watch`.

2. Verifique se o volume está montado em `/my/mount/path`.

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

3. Agora você pode excluir o Pod. O aplicativo Pod deixará de existir, mas o volume permanecerá.

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

Exemplos de manifestos

Manifestos de exemplo de PersistentVolumeClaim

Estes exemplos mostram opções básicas de configuração de PVC.

PVC com acesso RWO

Este exemplo mostra um PVC básico com acesso RWO associado a um StorageClass chamado `basic-csi`.

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

PVC com NVMe/TCP

Este exemplo mostra um PVC básico para NVMe/TCP com acesso RWO que está associado a um StorageClass chamado `protection-gold`.

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

Exemplos de manifesto de pod

Estes exemplos mostram configurações básicas para anexar o PVC a um pod.

Configuração básica

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

Configuração básica de NVMe/TCP

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

Consulte "Objetos Kubernetes e Trident" para obter detalhes sobre como as classes de armazenamento interagem com o `PersistentVolumeClaim` e os parâmetros para controlar como Trident provisiona volumes.

Expandir volumes

Trident oferece aos usuários do Kubernetes a capacidade de expandir seus volumes após eles serem criados. Encontre informações sobre as configurações necessárias para expandir volumes iSCSI, NFS, SMB, NVMe/TCP e FC.

Expandir um volume iSCSI

Você pode expandir um iSCSI Persistent Volume (PV) usando o provisionador CSI.

OBSERVAÇÃO	A expansão de volume iSCSI é suportada pelos `ontap-san`, `ontap-san-economy`, `solidfire-san` drivers e requer Kubernetes 1.16 ou posterior.

Passo 1: Configurar o StorageClass para suportar a expansão de volume

Edite a definição de StorageClass para definir o campo `allowVolumeExpansion` como `true`.

```
cat storageclass-ontapsan.yaml
```

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

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

Passo 2: Crie um PVC com o StorageClass que você criou

Edite a definição de PVC e atualize o `spec.resources.requests.storage` para refletir o novo tamanho desejado, que deve ser maior que o tamanho original.

```
cat pvc-ontapsan.yaml
```

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

Trident cria um Persistent Volume (PV) e o associa a este Persistent Volume Claim (PVC).

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

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

Etapa 3: defina um pod que conecte o PVC

Conecte o PV a um pod para que ele seja redimensionado. Existem dois cenários ao redimensionar um PV iSCSI:

- Se o PV estiver conectado a um pod, Trident expande o volume no backend de armazenamento, faz uma nova varredura no dispositivo e redimensiona o sistema de arquivos.
- Ao tentar redimensionar um PV não anexado, Trident expande o volume no backend de armazenamento. Depois que o PVC é vinculado a um pod, Trident verifica novamente o dispositivo e redimensiona o sistema de arquivos. O Kubernetes atualiza o tamanho do PVC após a conclusão bem-sucedida da operação de expansão.

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

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

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

Etapa 4: expandir o PV

Para redimensionar o PV criado de 1Gi para 2Gi, edite a definição do 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: valide a expansão

Você pode validar se a expansão funcionou corretamente conferindo o tamanho do PVC, PV e do volume do 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 FC

Você pode expandir um FC Persistent Volume (PV) usando o provisionador CSI.

OBSERVAÇÃO	A expansão de volume FC é suportada pelo `ontap-san` driver e requer Kubernetes 1.16 ou posterior.

Passo 1: Configurar o StorageClass para suportar a expansão de volume

Edite a definição de StorageClass para definir o campo `allowVolumeExpansion` como `true`.

```
cat storageclass-ontapsan.yaml
```

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

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

Passo 2: Crie um PVC com o StorageClass que você criou

Edite a definição de PVC e atualize o `spec.resources.requests.storage` para refletir o novo tamanho desejado, que deve ser maior que o tamanho original.

```
cat pvc-ontapsan.yaml
```

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

Trident cria um Persistent Volume (PV) e o associa a este Persistent Volume Claim (PVC).

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

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

Etapa 3: defina um pod que conecte o PVC

Conecte o PV a um pod para que ele seja redimensionado. Existem dois cenários ao redimensionar um PV FC:

- Se o PV estiver conectado a um pod, Trident expande o volume no backend de armazenamento, faz uma nova varredura no dispositivo e redimensiona o sistema de arquivos.
- Ao tentar redimensionar um PV não anexado, Trident expande o volume no backend de armazenamento. Depois que o PVC é vinculado a um pod, Trident verifica novamente o dispositivo e redimensiona o sistema de arquivos. O Kubernetes atualiza o tamanho do PVC após a conclusão bem-sucedida da

operação de expansão.

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

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

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

Etapa 4: expandir o PV

Para redimensionar o PV criado de 1Gi para 2Gi, edite a definição do 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: valide a expansão

Você pode validar se a expansão funcionou corretamente conferindo o tamanho do PVC, PV e do volume do 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

Trident suporta expansão de volume para NFS PVs provisionados em `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, e `azure-netapp-files` backends.

Passo 1: Configurar o StorageClass para suportar a expansão de volume

Para redimensionar um NFS PV, o administrador primeiro precisa configurar a classe de armazenamento para permitir expansão de volume, definindo o campo `allowVolumeExpansion` como `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, basta editar a classe de armazenamento

existente usando `kubectl edit storageclass` para permitir a expansão de volume.

Passo 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
```

Trident deve criar um NFS PV de 20 MiB 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
```

Etapa 3: expandir o PV

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

```
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: valide a expansão

Você pode verificar se o redimensionamento funcionou corretamente conferindo o tamanho do PVC, do PV e do Trident volume:

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

Entenda os limites do subsistema RWX NVMe

ReadWriteMany (RWX) volumes que utilizam o protocolo NVMe têm um limite de escalabilidade de 64 nós por volume. A seguir, são apresentadas as limitações, explicada a arquitetura do subsistema NVMe envolvido e descritas as etapas necessárias para a resolução do problema.

Entenda o limite de 64 nós

Se você planeja usar ReadWriteMany (RWX) volumes com o protocolo NVMe, um único volume RWX NVMe não pode ser montado por mais de 64 nós em um cluster Kubernetes.

Não agende cargas de trabalho que montem o mesmo RWX NVMe PersistentVolumeClaim em mais de 64 nós.

Essa limitação se aplica somente a volumes RWX que utilizam o protocolo NVMe.

Entenda os modelos de subsistema NVMe

Modelo de subsistema por volume (Trident releases anteriores a 26.02)

Nas versões do Trident anteriores à 26.02, os volumes RWX NVMe são provisionados usando um modelo de subsistema por volume. Cada volume RWX NVMe é mapeado para seu próprio subsistema NVMe dedicado no ONTAP.

Este modelo é simples, mas possui um limite de escalabilidade inferior. Em clusters Kubernetes de grande porte, os limites dos controladores de subsistema são atingidos rapidamente porque cada volume RWX consome um subsistema dedicado.

Modelo de super-subsistema (introduzido no Trident 26.02)

A partir do Trident 26.02, os volumes RWX NVMe utilizam um modelo de super-subsistema compartilhado. Vários volumes RWX NVMe compartilham o mesmo subsistema NVMe.

Cada super-subsistema suporta até 1024 namespaces (volumes). Esse modelo melhora significativamente a escalabilidade para cargas de trabalho RWX e reduz a probabilidade de atingir os limites do subsistema ONTAP.

Cada volume RWX NVMe suporta até 64 nós.

Identifique sintomas de erro

Se você criar ou anexar volumes RWX NVMe em grande escala, poderá observar erros semelhantes aos seguintes:

```
Maximum number of controllers reached. No more controllers can be created.
```

Este erro indica que o limite do controlador do subsistema NVMe do ONTAP foi atingido.

Resolver erros de limite do subsistema

Para superar as limitações de subsistemas por volume e aproveitar as vantagens do modelo de super-subsistema, atualize para Trident 26.02 ou posterior.

Atualize Trident para aplicar o modelo de super-subsistema

Para aplicar o modelo de super-subsistema para volumes RWX NVMe:

1. Atualize Trident para a versão 26.02 ou posterior.
2. Reduza para zero réplicas todos os pods que usam volumes RWX NVMe.
3. Verifique se nenhuma carga de trabalho está usando ativamente volumes RWX NVMe.
4. Aumente a escala dos pods novamente.

Essa sequência de reinicialização garante que os volumes RWX NVMe sejam conectados usando o modelo de super-subsistema.

- Essa limitação se aplica somente a volumes RWX que utilizam o protocolo NVMe.
- O limite de 64 nós se aplica por volume RWX NVMe.
- Outros modos de acesso e outros protocolos não são afetados.

Escalabilidade do controlador

Trident introduz escalabilidade do controlador por meio de maior simultaneidade entre vários drivers de armazenamento. Os clientes podem identificar quais drivers do Trident oferecem suporte à escalabilidade do controlador na disponibilidade geral e quais drivers estão disponíveis como prévia técnica no Trident 26.02. Isso permite decisões de implantação informadas e o gerenciamento adequado de riscos para ambientes Kubernetes escaláveis.

Conceitos e definições principais

Escalabilidade do controlador

A escalabilidade do controlador refere-se à capacidade do controlador Trident de processar múltiplas operações de armazenamento em paralelo, em vez de serializá-las por trás de um único bloqueio. Essas operações incluem criação, exclusão e redimensionamento de volumes, criação e exclusão de snapshots, publicação e cancelamento de publicação de volumes e gerenciamento de backend.

Quando a escalabilidade do controlador está habilitada, as operações em diferentes volumes e backends são executadas simultaneamente. Isso aumenta a taxa de transferência e reduz completamente o tempo de operação em ambientes com um grande número de operações simultâneas de PersistentVolumeClaim e VolumeSnapshot.

Suporte à escalabilidade do controlador

Trident oferece suporte à escalabilidade com diferentes níveis de maturidade, dependendo do driver de armazenamento.

Disponibilidade geral

Os seguintes drivers oferecem suporte à escalabilidade em disponibilidade geral no Trident 26.02:

- `ontap-san`
- `ontap-nas`
- `google-cloud-netapp-volumes`

OBSERVAÇÃO	Os `google-cloud-netapp-volumes` e `google-cloud-netapp-volumes-san` drivers são diferentes. Apenas `google-cloud-netapp-volumes` é suportado. Não use `google-cloud-netapp-volumes-san` em configurações de backend ou exemplos.

Habilitar escalabilidade do controlador

A escalabilidade do controlador é controlada pela `enableConcurrency` opção de configuração. Essa opção deve ser explicitamente habilitada durante a instalação do Trident ou atualizando uma implantação existente.

Desdobramento do operador Trident

Para habilitar a escalabilidade do controlador com o Trident operator, defina `enableConcurrency` como `true` no `TridentOrchestrator` recurso personalizado.

Nova instalação

Crie ou edite o `TridentOrchestrator` CR com `enableConcurrency` definido como `true`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableConcurrency: true
```

Aplique o CR:

```
kubectl apply -f tridentorchestrator_cr.yaml
```

Instalação existente

Corrija a `TridentOrchestrator`CR existente para permitir a escalabilidade do controlador:

```
kubectl patch torc trident --type=merge -p
'{"spec":{"enableConcurrency":true}}'
```

Verifique se a configuração foi aplicada:

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableConcurrency}'
```

Implantação Helm

Para habilitar a escalabilidade do controlador com o Helm, defina o valor `enableConcurrency` como `true`.

Nova instalação

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace --set enableConcurrency=true
```

Instalação existente

```
helm upgrade trident netapp-trident/trident-operator --namespace trident
--set enableConcurrency=true
```

Alternativamente, defina `enableConcurrency` para `true` em um arquivo `values.yaml` personalizado:

```
# values.yaml
enableConcurrency: true
```

Em seguida, instale ou atualize usando o arquivo de valores:

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace -f values.yaml
```

implantação do tridentctl

Para habilitar a escalabilidade do controlador com `tridentctl`, passe a flag `--enable-concurrency` durante a instalação.

Nova instalação

```
tridentctl install -n trident --enable-concurrency
```

Instalação existente

Para habilitar a escalabilidade do controlador em uma implantação existente baseada em `tridentctl`, desinstale e reinstale com a flag:

```
tridentctl uninstall -n trident
tridentctl install -n trident --enable-concurrency
```

Verifique se a escalabilidade está ativada

Após habilitar a escalabilidade do controlador, verifique se o Trident controller está em execução com a concorrência habilitada consultando os logs do pod do controlador:

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

Você deverá ver uma entrada de log indicando que a simultaneidade está habilitada.

Prévia técnica

Os seguintes drivers oferecem suporte à escalabilidade como uma prévia técnica no Trident 26.02:

- `nas-eco`
- `san-eco`

Para estes drivers:

- A concorrência de controladores está disponível para avaliação e teste
- O comportamento pode mudar em versões futuras
- O uso em ambientes de produção não é recomendado

Comportamento de concorrência

Quando a escalabilidade do controlador está ativada:

- Trident substitui o bloqueio global único por bloqueios granulares, por recurso
- Operações que modificam o mesmo recurso são serializadas para manter a consistência de dados
- Operações que apenas leem de um recurso podem prosseguir simultaneamente com outras operações de leitura nesse recurso
- Trident limita as solicitações simultâneas da API ONTAP a 20 por LIF de gerenciamento para evitar sobrecarga dos sistemas de armazenamento de backend
- Se vários backends compartilharem a mesma LIF de gerenciamento, eles compartilham esse limite de 20 solicitações

Limitações e considerações conhecidas

As seguintes considerações aplicam-se à escalabilidade do controlador:

- A concorrência é gerenciada internamente pelo controlador Trident
- Não há limites de simultaneidade configuráveis pelo usuário nesta versão
- A produtividade geral depende de:
  - O driver de armazenamento em uso
  - Responsividade do backend
  - Desempenho do servidor da API do Kubernetes
- Alta concorrência pode aumentar a carga nos sistemas de armazenamento de backend

Ressalvas e limitações

As seguintes limitações se aplicam em Trident 26.02:

- O comportamento de escalabilidade do controlador não é idêntico em todos os drivers
- Os drivers de pré-visualização técnica podem apresentar:
  - Desempenho inconsistente sob alta carga
  - Alterações de comportamento entre versões
- A depuração de operações simultâneas pode ser mais complexa devido à execução paralela
- As métricas e os logs podem mostrar saída de operações intercaladas

Recomendações

- Utilize drivers de disponibilidade geral (GA) para ambientes de produção que exigem alta escalabilidade
- Avalie drivers de pré-visualização técnica em ambientes de não produção
- Monitore o desempenho do backend e do controlador ao operar em escala

- Evite presumir a ordem das operações em scripts de automação

Expansão de volume automática

A expansão de volume permite que os Volumes Persistentes provisionados pelo Trident cresçam automaticamente quando a capacidade utilizada atinge um limite definido. Essa funcionalidade reduz a sobrecarga operacional e ajuda a evitar interrupções nos aplicativos causadas pelo esgotamento da capacidade. A expansão de volume automática é implementada usando Autogrow Policies. Uma Autogrow Policy define:

- O limite de utilização que desencadeia a expansão
- A quantidade pela qual o volume cresce
- O tamanho máximo que o volume pode atingir

OBSERVAÇÃO	Os volumes aumentam de tamanho automaticamente quando o limite de utilização definido é excedido. Os volumes nunca são reduzidos automaticamente.

Requisitos

Antes de configurar a expansão automática de volume, certifique-se de que os seguintes requisitos sejam atendidos:

- Trident 26.02 ou posterior
- Permissões de controle de acesso baseadas em funções para criar `TridentAutogrowPolicy` recursos personalizados
- StorageClasses configurado com `allowVolumeExpansion: true`
- Protocolos ONTAP suportados:
  - Network File System (NFS)
  - Internet Small Computer Systems Interface (iSCSI)
  - Memória Não Volátil Express (NVMe)
  - Protocolo Fibre Channel (FCP)

Limitações

- Os volumes de bloco bruto do ONTAP Non-Volatile Memory Express anteriores ao ONTAP 9.16.1 não suportam expansão automática.
- Para volumes de rede de área de armazenamento, se `growthAmount` for menor ou igual a 50 mebibytes, Trident aumenta automaticamente o valor para 51 mebibytes antes de redimensionar, desde que o tamanho resultante não exceda `maxSize`.
- Em ambientes já existentes, a expansão automática pode não funcionar para determinados volumes existentes devido ao comportamento de migração de publicação de volume.
- Quando um volume atinge `maxSize`, nenhuma expansão adicional ocorre.
- Protocolos suportados para expansão automática de volume:
  - Network File System (NFS)
  - Internet Small Computer Systems Interface (iSCSI)

- Memória Não Volátil Express (NVMe)
- Protocolo Fibre Channel (FCP)

Provisionar volumes com política de crescimento automático

A política de volume com crescimento automático pode ser configurada em dois níveis:

- Nível da classe de armazenamento: define o padrão para todos os volumes (usando anotação)
- Nível PVC: substitui o padrão da classe de armazenamento (usando anotação)

Criar uma política de crescimento automático

As políticas de crescimento automático permitem a expansão automática do volume quando os volumes atingem um limite de capacidade definido.

Certifique-se de ter:

- Trident 26.02 ou posterior instalado
- Permissões de controle de acesso baseadas em funções para criar `TridentAutogrowPolicy` recursos
- Compreensão dos requisitos de crescimento da carga de trabalho

Uma Autogrow Policy define como os volumes se expandem automaticamente quando atingem um limite de capacidade definido.

Você pode criar políticas de crescimento automático em qualquer ponto do seu fluxo de trabalho:

- Antes de StorageClasses e volumes serem criados
- Após StorageClasses existirem
- Após o provisionamento dos volumes

Essa flexibilidade permite que você introduza a expansão automática sem recriar os recursos existentes.

Especificações da política de volume com crescimento automático

As políticas de crescimento automático são recursos personalizados do Kubernetes definidos da seguinte forma:

Campo	Descrição	Formatar	Obrigatório	Exemplo	Padrão
nome	Identificador de política exclusivo	String	Sim	política-db-produção	Nenhum
usedThreshold	Percentual de capacidade que desencadeia a expansão	String de porcentagem	Sim	"80%"	Nenhum
growthAmount	Quantidade a crescer quando o limite for atingido	Porcentagem ou tamanho	Não	"10%" ou "5Gi"	"10%"

Campo	Descrição	Formatar	Obrigatório	Exemplo	Padrão
maxSize	Limite máximo de tamanho do volume	Quantidade de Kubernetes	Não	"500Gi"	Ilimitado

Criar uma política de crescimento automático

Passos

1. Crie um arquivo YAML que defina sua Autogrow Policy:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

2. Aplique a política ao seu cluster:

```
kubectl apply -f autogrow-policy.yaml
```

3. Verifique se a política foi criada:

```
kubectl get tridentautogrowpolicy standard-autogrow
```

Resultado esperado

```
NAME                USED THRESHOLD   GROWTH AMOUNT   STATE
standard-autogrow   80%              10%             Success
```

Estados de política

Depois que você cria uma política, Trident valida a especificação e atribui um dos seguintes estados:

Estado	Descrição	Ação necessária
Sucesso	A política é validada e está pronta para uso.	Nenhum.
Falha	Foram detectados erros de validação.	Revise e corrija a especificação.

Estado	Descrição	Ação necessária
Excluindo	A exclusão está em andamento.	Aguarde a conclusão.

Associe uma política a um StorageClass

Você pode associar uma Autogrow Policy a um StorageClass usando a `trident.netapp.io/autogrowPolicy` anotação. Todos os volumes provisionados a partir desse StorageClass herdam a política.

OBSERVAÇÃO O StorageClass deve ter `allowVolumeExpansion: true`.

Passos

1. Crie ou modifique uma StorageClass com a anotação de Autogrow Policy:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

2. Aplique o StorageClass:

```
kubectl apply -f storageclass.yaml
```

3. Verifique a anotação:

```
kubectl get storageclass ontap-gold -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Resultado esperado

```
production-db-policy
```

Precedência de política

Quando as anotações da Política de Crescimento Automático são definidas tanto em um StorageClass quanto em um PVC, Trident aplica as seguintes regras de precedência:

1. **A anotação PVC tem prioridade.** Se um PVC definir `trident.netapp.io/autogrowPolicy`, esse valor será sempre usado.
2. **StorageClass a anotação só se aplica quando o PVC não possui nenhuma anotação.**
3. **Se nenhum dos dois tiver a anotação,** nenhuma Autogrow Policy será aplicada.

StorageClass anotação	Anotação PVC	Comportamento eficaz
`trident.netapp.io/autogrowPolicy: standard-agp`	Não definido	Usos `standard-agp`.
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: logs-policy`	Usa `logs-policy` (PVC substitui StorageClass).
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: "none"`	Sem política de crescimento automático (PVC desativa o autogrow).
Não definido	`trident.netapp.io/autogrowPolicy: dev-policy`	Usos `dev-policy`.
Não definido	Não definido	Sem política de autogrow.

Exemplos de configuração

Os exemplos a seguir mostram configurações comuns de Autogrow Policy para diferentes casos de uso.

Política conservadora para bancos de dados de produção

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

Armazenamento de logs com incrementos de crescimento fixos

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: log-storage-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Política mínima com valores padrão

Quando você omite `growthAmount` e `maxSize`, Trident usa os padrões (`10%` de crescimento, tamanho ilimitado):

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: logs-policy
spec:
  usedThreshold: "85%"
```

Política com um maxSize personalizado e growthAmount padrão

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: default-ga-policy
spec:
  usedThreshold: "70%"
  maxSize: "100Gi"
```

Crescimento agressivo com maxSize ilimitado

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

Política com percentuais fracionários

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: precise-policy
spec:
  usedThreshold: "80.28%"
  growthAmount: "10.65%"
  maxSize: "100Gi"
```

OBSERVAÇÃO	São aceitas porcentagens fracionárias. Se você especificar mais de três casas decimais, Trident arredonda o valor para três casas decimais.

NAS StorageClass com Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN StorageClass com Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Gerenciar políticas de crescimento automático

Após criar as Autogrow Policies, você pode visualizá-las, atualizá-las e excluí-las conforme necessário. Você também pode monitorar quais volumes estão usando uma determinada policy.

Ver políticas de Autogrow

Listar todas as políticas

Use `kubectl` para listar todas as Autogrow Policies no seu cluster:

```
kubectl get tridentautogrowpolicy
```

Alternativamente, use `tridentctl`:

```
tridentctl get autogrowpolicy
```

Ver detalhes da política

Para visualizar a especificação completa e o status de uma policy:

```
kubectl describe tridentautogrowpolicy production-db-policy
```

Para visualizar uma apólice com seus respectivos volumes em formato YAML:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Atualizar uma política de crescimento automático

Você pode modificar uma política existente para alterar seu limite, quantidade de crescimento ou tamanho máximo. As alterações entram em vigor imediatamente para todos os volumes que utilizam a política.

IMPORTANTE	As alterações afetam todos os volumes que atualmente utilizam a política. Teste as alterações primeiro em um ambiente que não seja de produção, sempre que possível.

Passos

1. Edite a política:

   ```
   kubectl edit tridentautogrowpolicy production-db-policy
   ```

2. Modifique os `spec` fields conforme necessário:

```
spec:
  usedThreshold: "75%"     # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

3. Salve e saia. As alterações entram em vigor imediatamente.

Considerações sobre atualização

- **Efeito imediato:** Todos os volumes que utilizam a política adotam os novos parâmetros na próxima avaliação de crescimento.
- **Não é necessário reiniciar o volume:** As alterações serão aplicadas na próxima operação de crescimento.
- **Teste primeiro:** Valide as alterações em um ambiente que não seja de produção quando possível.
- **Comunique alterações:** Notifique as equipes quando você modificar as políticas compartilhadas.

Excluir uma política de crescimento automático

As políticas de crescimento automático usam proteção de finalizador para evitar exclusão acidental enquanto os volumes estão usando-as ativamente.

Passos

1. Excluir a política:

```
kubectl delete tridentautogrowpolicy production-db-policy
```

2. Se os volumes ainda estiverem usando a política, a exclusão entra em um `Deleting` estado. Verifique quais volumes são afetados:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

3. Remova a política de cada volume afetado. Escolha uma das seguintes opções:

   - **Opção A: desative explicitamente o crescimento automático** definindo a anotação como `"none"`:

   ```
   kubectl annotate pvc <pvc-name> \
     trident.netapp.io/autogrowPolicy="none" \
     --overwrite
   ```

   - **Opção B: remover a anotação completamente**

   ```
   kubectl annotate pvc <pvc-name> \
     trident.netapp.io/autogrowPolicy-
   ```

Comportamento de exclusão

Cenário	Comportamento
Nenhum volume utiliza a política	A política é excluída imediatamente.
Os volumes estão utilizando a política	A política entra em `Deleting` estado. Um finalizador bloqueia a conclusão até que todos os volumes sejam removidos.
Todos os volumes são removidos da política	Os finalizadores são removidos e a policy é excluída.

Monitorar o uso da política Autogrow

Verificar volumes usando uma política

```
tridentctl get autogrowpolicy production-db-policy -o json | jq '.volumes'
```

Descubra qual política um volume utiliza

```
kubectl get pvc database-pvc -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Monitorar eventos de políticas

```
kubectl get events --field-selector
involvedObject.kind=TridentAutogrowPolicy
```

Protocolos suportados

Autogrow suporta os seguintes protocolos de armazenamento:

- NFS
- iSCSI
- FCP
- NVMe

OBSERVAÇÃO	Para volumes SAN, se o `growthAmount` configurado for de 50 MiB ou menos, Trident aumenta automaticamente o valor de crescimento para 51 MB na operação de redimensionamento, desde que o tamanho resultante não exceda `maxSize`.

Limitações conhecidas

- **ONTAP Volumes de bloco bruto NVMe:** Volumes criados com versões do ONTAP anteriores à 9.16.1 não suportam autogrow.
- **Volumes existentes (implantações brownfield):** O Autogrow pode não funcionar para volumes existentes, mesmo que uma Política de Autogrow válida seja aplicada. Isso ocorre devido a uma migração

em andamento das publicações de volume. Para confirmar se a migração foi concluída, verifique os logs do controlador Trident para mensagens "`Migration completed`".

Perguntas frequentes

Quando o Trident avalia o limiar?

Trident monitora continuamente o uso do volume. Quando a capacidade utilizada ultrapassa o `usedThreshold`, Trident cria uma solicitação interna de redimensionamento e expande o volume pela configuração de `growthAmount`.

Por exemplo, esta política aciona a expansão em 80% da capacidade e aumenta o volume em 10% cada vez, até um máximo de 500 GiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

Posso aplicar uma política depois que os volumes já foram provisionados?

Sim. Você pode criar uma Política de Crescimento Automático a qualquer momento e aplicá-la a PVCs existentes adicionando ou atualizando a `trident.netapp.io/autogrowPolicy` anotação. Você não precisa recriar o PVC ou o StorageClass.

Para aplicar uma política a um PVC existente:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Para aplicar uma política a um StorageClass existente:

```
kubectl annotate storageclass ontap-gold \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

O que acontece se eu definir uma Autogrow Policy tanto no StorageClass quanto no PVC?

A anotação do PVC sempre tem precedência. Se um PVC tiver a `trident.netapp.io/autogrowPolicy` anotação, Trident usa esse valor independentemente do que o StorageClass especifica. Consulte "Precedência de política" para obter detalhes.

Por exemplo, dado este StorageClass:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

E este PVC que se sobrepõe à política de StorageClass:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
  storageClassName: ontap-gold
```

Trident usa `logs-policy` para `database-pvc`, não `standard-agp`.

Como faço para desativar o autogrow para um volume específico?

Defina a anotação PVC como `"none"`. Isso substitui qualquer política de nível de StorageClass para esse volume:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

Você pode verificar se o autogrow está desativado:

```
kubectl get pvc <pvc-name> -o jsonpath
='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Resultado esperado

```
none
```

O que acontece quando um volume atinge maxSize?

Trident interrompe a expansão do volume. Nenhuma outra solicitação de redimensionamento é criada para esse volume, mesmo que o uso continue a aumentar além do `usedThreshold`.

Por exemplo, com essa política, Trident para de crescer o volume assim que ele atinge 100 GiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: capped-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Para permitir crescimento ilimitado, omita `maxSize` ou defina como `0`:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: unlimited-policy
spec:
  usedThreshold: "85%"
  growthAmount: "10%"
```

Posso alterar uma política sem reiniciar os volumes?

Sim. Quando você atualiza uma política, todos os volumes que usam essa política adotam os novos parâmetros na próxima avaliação de crescimento. Não é necessário reiniciar nenhum volume.

Para atualizar uma política existente:

```
kubectl edit tridentautogrowpolicy production-db-policy
```

Modifique os campos conforme necessário:

```
spec:
  usedThreshold: "75%"    # Changed from 80%
  growthAmount: "20%"     # Changed from 10%
  maxSize: "1Ti"          # Changed from 500Gi
```

Salve e saia. Verifique a política atualizada:

```
kubectl get tridentautogrowpolicy production-db-policy
```

Resultado esperado

```
NAME                    USED THRESHOLD   GROWTH AMOUNT   STATE
production-db-policy    75%              20%             Success
```

Por que minha policy está em estado de falha?

Um `Failed` estado indica que a especificação da política contém erros de validação. Execute o seguinte comando para visualizar os detalhes do erro:

```
kubectl describe tridentautogrowpolicy <policy-name>
```

As causas comuns incluem um `usedThreshold` inválido (deve estar entre 1–99%), um `growthAmount` que excede `maxSize`, ou um formato de quantidade do Kubernetes inválido. Corrija a especificação e reaplique:

```
kubectl apply -f autogrow-policy.yaml
```

Por que não consigo excluir uma política?

As políticas utilizam proteção de finalização. Se os volumes ainda estiverem utilizando a política, a exclusão entra em um estado `Deleting` e aguarda até que todos os volumes sejam removidos da política.

Identifique os volumes afetados:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Em seguida, remova a anotação de cada PVC:

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Depois que todos os volumes são removidos, o finalizador é liberado e a política é excluída.

O recurso de crescimento automático funciona com todos os backends do ONTAP?

Autogrow é compatível com os protocolos NFS, iSCSI, FCP e NVMe. No entanto, volumes de bloco bruto NVMe exigem ONTAP 9.16.1 ou posterior.

Os volumes existentes em implantações brownfield podem exigir que a migração da publicação de volume seja concluída antes que o autogrow entre em vigor. Verifique o status da migração consultando os logs do controlador Trident:

```
kubectl logs -l app=trident-controller -n trident | grep "Migration
completed"
```

Os seguintes exemplos de StorageClass mostram o autogrow configurado para backends NAS e SAN:

Backend NAS

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

Backend SAN

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Qual é o valor mínimo de crescimento para volumes SAN?

Para volumes SAN, o crescimento mínimo efetivo é de 51 MB. Se você configurar um `growthAmount` de 50 MiB ou menos, Trident aumenta automaticamente o crescimento para 51 MB na operação de redimensionamento.

Por exemplo, esta política define um `growthAmount` de `"40Mi"`, mas Trident aplica um crescimento de 51 MB para qualquer volume SAN que a utilize:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-minimal-policy
spec:
  usedThreshold: "85%"
  growthAmount: "40Mi"
  maxSize: "100Gi"
```

Para evitar esse ajuste automático, defina `growthAmount` para um valor superior a 50 MiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-policy
spec:
  usedThreshold: "85%"
  growthAmount: "100Mi"
  maxSize: "500Gi"
```

Importar volumes

Você pode importar volumes de armazenamento existentes como um PV do Kubernetes usando `tridentctl import` ou criando uma Persistent Volume Claim (PVC) com anotações de importação do Trident.

Visão geral e considerações

Você pode importar um volume para Trident para:

- Containerize uma aplicação e reutilize seu conjunto de dados existente
- Utilize um clone de um conjunto de dados para uma aplicação efêmera
- Reconstruir um cluster Kubernetes com falha
- Migrar dados de aplicativo durante a recuperação de desastres

Considerações

Antes de importar um volume, revise as seguintes considerações.

- Trident pode importar apenas volumes ONTAP do tipo RW (leitura e gravação). Volumes do tipo DP (proteção de dados) são volumes de destino do SnapMirror. Você deve quebrar a relação de espelhamento antes de importar o volume para Trident.
- Sugerimos importar volumes sem conexões ativas. Para importar um volume em uso ativo, clone o volume e depois execute a importação.

AVISO	Isso é especialmente importante para volumes de bloco, pois o Kubernetes desconheceria a conexão anterior e poderia facilmente anexar um volume ativo a um pod. Isso pode resultar em corrupção de dados.

- Embora `StorageClass` deva ser especificado em um PVC, Trident não utiliza esse parâmetro durante a importação. As classes de armazenamento são usadas durante a criação do volume para selecionar entre os pools disponíveis com base nas características de armazenamento. Como o volume já existe, nenhuma seleção de pool é necessária durante a importação. Portanto, a importação não falhará mesmo que o volume exista em um backend ou pool que não corresponda à classe de armazenamento especificada no PVC.
- O tamanho do volume existente é determinado e definido no PVC. Após o volume ser importado pelo driver de armazenamento, o PV é criado com um ClaimRef para o PVC.
  - A política de recuperação é inicialmente definida como `retain` no PV. Depois que o Kubernetes vincula com sucesso o PVC e o PV, a política de recuperação é atualizada para corresponder à política de recuperação da Storage Class.
  - Se a política de recuperação da Storage Class for `delete`, o volume de armazenamento será excluído quando o PV for excluído.
- Por padrão, Trident gerencia o PVC e renomeia o FlexVol volume e o LUN no backend. Você pode usar a `--no-manage` flag para importar um volume não gerenciado e a `--no-rename` flag para manter o nome do volume.
  - `--no-manage`* - Se você usar a `--no-manage` flag, Trident não executa nenhuma operação adicional no PVC ou PV para o ciclo de vida dos objetos. O volume de armazenamento não é excluído quando o PV é excluído e outras operações, como clone de volume e expansão de volume, também são ignoradas.

◦ `--no-rename*` - Se você usar a `--no-rename`flag, Trident mantém o nome do volume existente ao importar volumes e gerencia o ciclo de vida dos volumes. Essa opção é compatível apenas para os drivers `ontap-nas`, `ontap-san` (incluindo sistemas ASA r2) e `ontap-san-economy`.

DICA	Essas opções são úteis se você deseja usar o Kubernetes para cargas de trabalho em contêineres, mas de outra forma deseja gerenciar o ciclo de vida do volume de armazenamento fora do Kubernetes.

- Uma anotação é adicionada ao PVC e ao PV com a dupla função de indicar que o volume foi importado e se o PVC e o PV são gerenciados. Essa anotação não deve ser modificada ou removida.

Importar um volume

Você pode importar um volume usando `tridentctl import` ou criando um PVC com anotações de importação do Trident.

OBSERVAÇÃO	Se você utiliza anotações PVC, não precisa baixar ou usar `tridentctl` para importar o volume.

Usando tridentctl

Passos

1. Crie um arquivo PVC (por exemplo, `pvc.yaml`) que será usado para criar o PVC. O arquivo PVC deve incluir `name`, `namespace`, `accessModes` e `storageClassName`. Opcionalmente, você pode especificar `unixPermissions` na sua definição de PVC.

   A seguir está um exemplo de especificação mínima:

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

OBSERVAÇÃO	Inclua apenas os parâmetros necessários. Parâmetros adicionais, como o nome do PV ou o tamanho do volume, podem fazer com que o comando de importação falhe.

2. Use o `tridentctl import` comando para especificar o nome do backend Trident que contém o volume e o nome que identifica exclusivamente o volume no armazenamento (por exemplo: ONTAP FlexVol, Element Volume). O `-f` argumento é obrigatório para especificar o caminho para o arquivo PVC.

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

Usando anotações em PVC

Passos

1. Crie um arquivo YAML de PVC (por exemplo, `pvc.yaml`) com as anotações de importação do Trident necessárias. O arquivo de PVC deve incluir:

   - `name` e `namespace` em metadados
   - `accessModes`, `resources.requests.storage`, e `storageClassName` em especificação
   - Anotações:
     - `trident.netapp.io/importOriginalName`: Nome do volume no backend
     - `trident.netapp.io/importBackendUUID`: UUID do backend onde o volume existe
     - `trident.netapp.io/notManaged` (*Opcional*): Defina como `"true"` para volumes não gerenciados. O padrão é `"false"`.

A seguir está um exemplo de especificação para importar um volume gerenciado:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. Aplique o arquivo YAML do PVC ao seu cluster Kubernetes:

```
kubectl apply -f <pvc-file>.yaml
```

Trident importará automaticamente o volume e o vinculará ao PVC.

Exemplos

Revise os seguintes exemplos de importação de volume para drivers compatíveis.

ONTAP NAS e ONTAP NAS FlexGroup

Trident suporta importação de volume usando os `ontap-nas` e `ontap-nas-flexgroup` drivers.

OBSERVAÇÃO

- Trident não oferece suporte à importação de volumes usando o `ontap-nas-economy` driver.
- Os `ontap-nas` e `ontap-nas-flexgroup` drivers não permitem nomes de volume duplicados.

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

Exemplos de ONTAP NAS usando tridentctl

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados usando `tridentctl`.

Volume gerenciado

O exemplo a seguir importa um volume chamado `managed_volume` em um backend chamado `ontap_nas`:

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

Volume não gerenciado

Ao usar o `--no-manage` argumento, Trident não renomeia o volume.

O exemplo a seguir importa `unmanaged_volume`no `ontap_nas`backend:

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

Exemplos do ONTAP NAS usando anotações PVC

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados usando anotações PVC.

Volume gerenciado

O exemplo a seguir importa um volume de 1GiB `ontap-nas` nomeado `ontap_volume1` do backend `81abcb27-ea63-49bb-b606-0a5315ac5f21` com o modo de acesso RWO definido usando anotações PVC:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

Volume não gerenciado

O exemplo a seguir importa 1Gi `ontap-nas` volume nomeado `ontap-volume2` do backend `34abcb27-ea63-49bb-b606-0a5315ac5f34` com o modo de acesso RWO definido usando anotações PVC:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <umanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-
0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

ONTAP SAN

Trident suporta a importação de volumes usando os `ontap-san` (iSCSI, NVMe/TCP e FC) e `ontap-san-economy`drivers.

Trident pode importar volumes ONTAP SAN FlexVol que contêm um único LUN. Isso é consistente com o `ontap-san` driver, que cria um volume FlexVol para cada PVC e um LUN dentro do volume FlexVol. Trident importa o volume FlexVol e o associa à definição do PVC. Trident pode importar `ontap-san-economy` volumes que contêm múltiplos LUNs.

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados:

Volume gerenciado

Para volumes gerenciados, Trident renomeia o FlexVol volume para o `pvc-<uuid>` formato e o LUN dentro do FlexVol volume para `lun0`.

O exemplo a seguir importa o `ontap-san-managed` FlexVol volume que está presente no `ontap_san_default` backend:

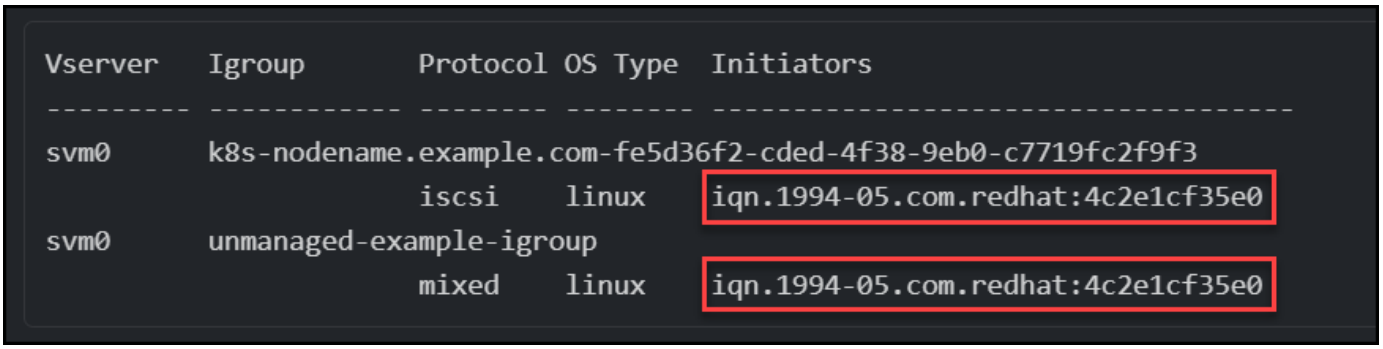
```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

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

Volume não gerenciado

O exemplo a seguir importa `unmanaged_example_volume`no `ontap_san`backend:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

Se você tiver LUNs mapeadas para igroups que compartilham um IQN com o IQN de um nó do Kubernetes, como mostrado no exemplo a seguir, você receberá o erro: `LUN already mapped to initiator(s) in this group`. Você precisará remover o iniciador ou desmapear a LUN para importar o volume.



Exemplos de SAN-economy do ONTAP

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados para o `ontap-san-economy` backend.

Volume gerenciado

Ao importar um volume gerenciado, Trident assume a propriedade do FlexVol e o renomeia. Você deve levar em consideração essa renomeação ao importar vários LUNs do mesmo FlexVol.

O exemplo a seguir importa `lun1` de FlexVol `toimport` como um volume gerenciado chamado `vol-managed-saneco`:

```
tridentctl import volume vol-managed-saneco toimport/lun1 -f
import1.yaml
```

Após a importação `lun1`, Trident renomeia o FlexVol (por exemplo, para `trident_lun_pool_xyz`). Para importar LUNs adicionais do mesmo FlexVol, use o novo nome do FlexVol:

```
tridentctl import volume vol-managed-saneco trident_lun_pool_xyz/lun2
-f import2.yaml
```

OBSERVAÇÃO	O `ontap-san-economy` backend importa um LUN por vez. Você pode automatizar várias importações usando um script.

Volume não gerenciado

Ao importar um volume não gerenciado, Trident não assume a propriedade do FlexVol. No entanto, o FlexVol e o LUN devem seguir as convenções de nomenclatura do Trident.

Formato de nomenclatura do FlexVol

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- `STORAGEPREFIX` é o valor de `storagePrefix` na sua configuração de backend. O padrão é `trident`.
- `RANDOMSTRING` é qualquer sequência de caracteres que você escolher.

Requisito de nomenclatura LUN

O LUN deve ser nomeado `lun0`.

Exemplo

Se o seu `storagePrefix` é `xyz`, o caminho completo para o LUN é:

```
trident_lun_pool_xyz_randomstring/lun0
```

Elemento

Trident oferece suporte ao software NetApp Element e à importação de volumes NetApp HCI usando o driver `solidfire-san`.

OBSERVAÇÃO	O driver Element suporta nomes de volume duplicados. No entanto, Trident retorna um erro se houver nomes de volume duplicados. Como solução alternativa, clone o volume, forneça um nome de volume exclusivo e importe o volume clonado.

O exemplo a seguir importa um `element-managed` volume no backend `element_default`.

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

Azure NetApp Files

Trident oferece suporte à importação de volumes usando o `azure-netapp-files` driver.

OBSERVAÇÃO	Para importar um volume do Azure NetApp Files, identifique o volume pelo seu caminho de volume. O caminho do volume é a parte do caminho de exportação do volume após o `:/`. Por exemplo, se o caminho de montagem for `10.0.0.2:/importvol1`, o caminho do volume é `importvol1`.

O exemplo a seguir importa um `azure-netapp-files` volume no backend `azurenetappfiles_40517` com o caminho do volume `importvol1`.

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

Google Cloud NetApp Volumes

Trident oferece suporte à importação de volumes usando o `google-cloud-netapp-volumes` driver.

O exemplo a seguir importa um volume no backend `backend-tbc-gcnv1` com o volume `testvoleasiaeast1`.

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

O exemplo a seguir importa um `google-cloud-netapp-volumes` volume quando dois volumes estão presentes na mesma região:

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

Personalize os nomes e rótulos dos volumes

Com Trident, você pode atribuir nomes e rótulos significativos aos volumes que cria. Isso ajuda você a identificar e mapear facilmente os volumes aos seus respectivos recursos do Kubernetes (PVCs). Você também pode definir modelos no nível do backend para criar nomes de volumes personalizados e rótulos personalizados; quaisquer volumes que você criar, importar ou clonar seguirão os modelos.

Antes de começar

Suporte para nomes e rótulos de volume personalizáveis:

- Operações de criação, importação e clonagem de volume.
- No caso do `ontap-nas-economy`driver, apenas o nome do volume Qtree está em conformidade com o modelo de nome.
- No caso do `ontap-san-economy`driver, apenas o nome do LUN está em conformidade com o modelo de nome.

Limitações

- Os nomes de volumes personalizados são compatíveis apenas com drivers ONTAP locais.
- Rótulos personalizados são suportados apenas para os `ontap-san`, `ontap-nas` e `ontap-nas-flexgroup` drivers.
- Nomes de volume personalizados não se aplicam a volumes existentes.

Principais comportamentos dos nomes de volume personalizáveis

- Se ocorrer uma falha devido à sintaxe inválida em um modelo de nome, a criação do backend falhará. No entanto, se a aplicação do modelo falhar, o volume será nomeado de acordo com a convenção de nomenclatura existente.
- O prefixo de armazenamento não se aplica quando um volume é nomeado usando um modelo de nome da configuração do backend. Qualquer valor de prefixo desejado pode ser adicionado diretamente ao modelo.

Exemplos de configuração de backend com template de nome e rótulos

É possível definir modelos de nomes personalizados no nível raiz e/ou no nível do pool.

Exemplo de nível raiz

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.Requ
estName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

Exemplo de nível de pool

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels": {
        "labelname": "label1",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels": {
        "cluster": "label2",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

Exemplos de modelos de nome

Exemplo 1:

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

Exemplo 2:

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

Pontos a considerar

1. No caso de importações de volume, os rótulos são atualizados somente se o volume existente possuir rótulos em um formato específico. Por exemplo: `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. No caso de importações de volumes gerenciados, o nome do volume segue o modelo de nome definido no nível raiz na definição do backend.
3. Trident não suporta o uso de um operador de slice com o prefixo de armazenamento.
4. Se os modelos não resultarem em nomes de volume exclusivos, Trident adicionará alguns caracteres aleatórios para criar nomes de volume exclusivos.
5. Se o nome personalizado para um volume econômico do NAS exceder 64 caracteres, Trident nomeará os volumes de acordo com a convenção de nomenclatura existente. Para todos os outros drivers ONTAP, se o nome do volume exceder o limite de nomes, o processo de criação do volume falhará.

Compartilhe um volume NFS entre namespaces

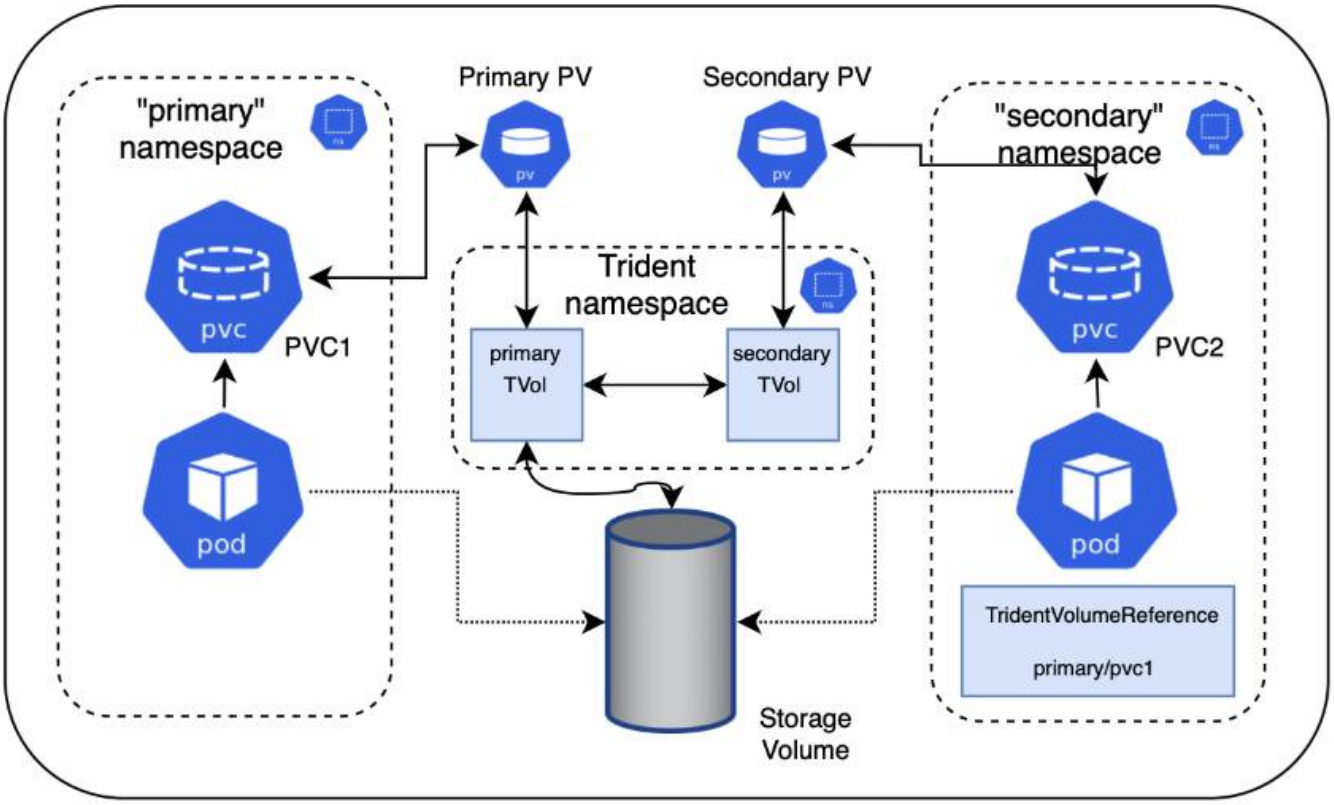
Usando Trident, você pode criar um volume em um namespace primário e compartilhá-lo em um ou mais namespaces secundários.

Características

O CR TridentVolumeReference permite que você compartilhe com segurança volumes NFS ReadWriteMany (RWX) entre um ou mais namespaces do Kubernetes. Essa solução nativa do Kubernetes oferece os seguintes benefícios:

- Múltiplos níveis de controle de acesso para garantir a segurança
- Compatível com todos os drivers de volume Trident NFS
- Sem dependência do tridentctl ou de qualquer outro recurso não nativo do Kubernetes

Este diagrama ilustra o compartilhamento de volumes NFS entre dois namespaces do Kubernetes.



Início rápido

Você pode configurar o compartilhamento de volumes NFS em apenas alguns passos.

1 **Configure o PVC de origem para compartilhar o volume**

O proprietário do namespace de origem concede permissão para acessar os dados no PVC de origem.

2 **Conceda permissão para criar um CR no namespace de destino**

O administrador do cluster concede permissão ao proprietário do namespace de destino para criar o TridentVolumeReference CR.

3 **Criar TridentVolumeReference no namespace de destino**

O proprietário do namespace de destino cria o TridentVolumeReference CR para fazer referência ao PVC de origem.

4 **Crie o PVC subordinado no namespace de destino**

O proprietário do namespace de destino cria o PVC subordinado para usar a fonte de dados do PVC de origem.

Configure os namespaces de origem e destino

Para garantir a segurança, o compartilhamento entre namespaces exige colaboração e ação do proprietário do namespace de origem, do administrador do cluster e do proprietário do namespace de destino. A função do usuário é designada em cada etapa.

Passos

1. **Proprietário do namespace de origem:** Crie o PVC (`pvc1` no namespace de origem que concede permissão para compartilhar com o namespace de destino (`namespace2` usando a anotação `shareToNamespace`.

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

   Trident cria o PV e seu volume de armazenamento NFS de backend.

OBSERVAÇÃO

- Você pode compartilhar o PVC com vários namespaces usando uma lista separada por vírgulas. Por exemplo, `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4`.
- Você pode compartilhar com todos os namespaces usando `*`. Por exemplo, `trident.netapp.io/shareToNamespace: *`
- Você pode atualizar o PVC para incluir a `shareToNamespace` anotação a qualquer momento.

2. **Administrador do cluster:** Certifique-se de que o RBAC adequado esteja implementado para conceder permissão ao proprietário do namespace de destino para criar o CR TridentVolumeReference no namespace de destino.
3. **Proprietário do namespace de destino:** Crie um CR TridentVolumeReference no namespace de destino que faça referência ao namespace de origem `pvc1`.

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

4. **Proprietário do namespace de destino:** Crie um PVC (`pvc2`) no namespace de destino (`namespace2`) usando a anotação `shareFromPVC` para designar o PVC de origem.

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

OBSERVAÇÃO	O tamanho do PVC de destino deve ser menor ou igual ao do PVC de origem.

Resultados

Trident lê a `shareFromPVC` anotação no PVC de destino e cria o PV de destino como um volume subordinado, sem recurso de armazenamento próprio, que aponta para o PV de origem e compartilha o recurso de armazenamento do PV de origem. O PVC de destino e o PV de destino aparecem vinculados normalmente.

Excluir um volume compartilhado

Você pode excluir um volume compartilhado entre vários namespaces. Trident removerá o acesso ao volume no namespace de origem e manterá o acesso para outros namespaces que compartilham o volume. Quando todos os namespaces que fazem referência ao volume são removidos, Trident exclui o volume.

Use `tridentctl get` para consultar volumes subordinados

Utilizando o [`tridentctl` utilitário, você pode executar o `get` comando para obter volumes subordinados. Para mais informações, consulte `tridentctl` comandos e opções.

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

Bandeiras:

- `-h, --help`: Ajuda para volumes.
- `--parentOfSubordinate string`: limitar a consulta ao volume de origem subordinado.
- `--subordinateOf string`: limitar a consulta aos subordinados do volume.

Limitações

- Trident não pode impedir que os namespaces de destino gravem no volume compartilhado. Você deve usar bloqueio de arquivos ou outros processos para evitar a sobrescrita de dados do volume compartilhado.
- Não é possível revogar o acesso ao PVC de origem removendo as `shareToNamespace` ou `shareFromNamespace` anotações ou excluindo o `TridentVolumeReference` CR. Para revogar o acesso, é necessário excluir o PVC subordinado.
- Snapshots, clones e espelhamento não são possíveis em volumes subordinados.

Para obter mais informações

Para saber mais sobre acesso a volumes entre namespaces:

- Veja a demonstração em "NetAppTV".

Clonar volumes entre namespaces

Usando Trident, você pode criar novos volumes usando volumes existentes ou volumesnapshots de um namespace diferente dentro do mesmo cluster Kubernetes.

Pré-requisitos

Antes de clonar volumes, certifique-se de que os backends de origem e destino sejam do mesmo tipo e tenham a mesma classe de armazenamento.

OBSERVAÇÃO	A clonagem entre namespaces é suportada apenas para os `ontap-san` e `ontap-nas` drivers de armazenamento. Clones somente leitura não são suportados.

Início rápido

Você pode configurar a clonagem de volumes em apenas alguns passos.

1. **Configure o PVC de origem para clonar o volume**

O proprietário do namespace de origem concede permissão para acessar os dados no PVC de origem.

2 Conceda permissão para criar um CR no namespace de destino

O administrador do cluster concede permissão ao proprietário do namespace de destino para criar o TridentVolumeReference CR.

3 Criar TridentVolumeReference no namespace de destino

O proprietário do namespace de destino cria o TridentVolumeReference CR para fazer referência ao PVC de origem.

4 Crie o PVC clonado no namespace de destino

O proprietário do namespace de destino cria um PVC para clonar o PVC do namespace de origem.

Configure os namespaces de origem e destino

Para garantir a segurança, a clonagem de volumes entre namespaces requer colaboração e ação do proprietário do namespace de origem, do administrador do cluster e do proprietário do namespace de destino. A função do usuário é designada em cada etapa.

Passos

1. **Proprietário do namespace de origem:** Crie o PVC (`pvc1`) no namespace de origem (`namespace1`) que concede permissão para compartilhar com o namespace de destino (`namespace2`) usando a anotação `cloneToNamespace`.

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

   Trident cria o PV e seu volume de armazenamento backend.

OBSERVAÇÃO

- Você pode compartilhar o PVC com vários namespaces usando uma lista separada por vírgulas. Por exemplo, `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4`.
- Você pode compartilhar com todos os namespaces usando `*`. Por exemplo, `trident.netapp.io/cloneToNamespace: *`
- Você pode atualizar o PVC para incluir a `cloneToNamespace` anotação a qualquer momento.

2. **Administrador do cluster:** Certifique-se de que o RBAC adequado esteja configurado para conceder permissão ao proprietário do namespace de destino para criar o CR TridentVolumeReference no namespace de destino (`namespace2`).
3. **Proprietário do namespace de destino:** Crie um CR TridentVolumeReference no namespace de destino que faça referência ao namespace de origem `pvc1`.

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

4. **Proprietário do namespace de destino:** Crie um PVC (`pvc2`) no namespace de destino (`namespace2`) usando as `cloneFromPVC` ou `cloneFromSnapshot`, e `cloneFromNamespace` anotações para designar o PVC de origem.

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

Limitações

- Para PVCs provisionados usando drivers ontap-nas-economy, clones somente leitura não são suportados.

Replicar volumes usando SnapMirror

Trident oferece suporte a relações de espelhamento entre um volume de origem em um cluster e o volume de destino no cluster emparelhado para replicação de dados em casos de recuperação de desastres. Você pode usar uma Definição de Recurso Personalizado (CRD) com namespace, chamada Trident Mirror Relationship (TMR), para realizar as seguintes operações:

- Criar relações de espelhamento entre volumes (PVCs)
- Remova relações de espelhamento entre volumes
- Interrompa os relacionamentos espelhados
- Promover o volume secundário em situações de desastre (falhas)
- Realizar a transição sem perda de dados de aplicações de um cluster para outro (durante failovers ou migrações planejadas)

Pré-requisitos de replicação

Certifique-se de que os seguintes pré-requisitos sejam atendidos antes de começar:

Clusters do ONTAP

- **Trident**: A versão 22.10 ou posterior do Trident deve existir em ambos os clusters Kubernetes de origem e destino que utilizam ONTAP como backend.
- **Licenças**: As licenças assíncronas do ONTAP SnapMirror usando o pacote Data Protection devem estar habilitadas tanto nos clusters ONTAP de origem quanto de destino. Consulte "SnapMirror visão geral do licenciamento no ONTAP" para mais informações.

  A partir do ONTAP 9.10.1, todas as licenças são fornecidas como um arquivo de licença NetApp (NLF), que é um único arquivo que habilita vários recursos. Consulte "Licenças incluídas com o ONTAP One" para mais informações.

OBSERVAÇÃO	Apenas a proteção assíncrona SnapMirror é suportada.

Peering

- **Cluster e SVM**: Os backends de armazenamento ONTAP devem estar emparelhados. Consulte "Visão geral do peering de cluster e SVM" para mais informações.

IMPORTANTE	Certifique-se de que os nomes SVM usados na relação de replicação entre dois clusters ONTAP sejam únicos.

- **Trident e SVM**: Os SVMs remotos emparelhados devem estar disponíveis para Trident no cluster de destino.

Drivers com suporte

NetApp Trident suporta replicação de volumes com a tecnologia NetApp SnapMirror usando classes de armazenamento com suporte dos seguintes drivers: **`ontap-nas`: NFS** `ontap-san`: iSCSI **`ontap-san`: FC**

`ontap-san`: NVMe/TCP (requer versão mínima do ONTAP 9.15.1)

OBSERVAÇÃO	A replicação de volume usando SnapMirror não é suportada para sistemas ASA r2. Para obter informações sobre sistemas ASA r2, consulte "Saiba mais sobre os sistemas de armazenamento ASA r2".

Crie um PVC espelhado

Siga estes passos e utilize os exemplos de CRD para criar relação de espelhamento entre os volumes primário e secundário.

Passos

1. Execute as seguintes etapas no cluster Kubernetes primário:
   a. Crie um objeto StorageClass com o parâmetro `trident.netapp.io/replication: true`.

      Exemplo

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        backendType: "ontap-nas"
        fsType: "nfs"
        trident.netapp.io/replication: "true"
      ```

   b. Crie um PVC com StorageClass previamente criado.

      Exemplo

      ```
      kind: PersistentVolumeClaim
      apiVersion: v1
      metadata:
        name: csi-nas
      spec:
        accessModes:
        - ReadWriteMany
        resources:
          requests:
            storage: 1Gi
        storageClassName: csi-nas
      ```

   c. Crie um MirrorRelationship CR com informações locais.

Exemplo

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Trident obtém as informações internas do volume e o estado atual de proteção de dados (DP) do volume, em seguida, preenche o campo de status do MirrorRelationship.

d. Obtenha o TridentMirrorRelationship CR para obter o nome interno e o SVM do PVC.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. Execute as seguintes etapas no cluster Kubernetes secundário:

   a. Crie um StorageClass com o parâmetro trident.netapp.io/replication: true.

Exemplo

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/replication: true
```

b. Crie um CR MirrorRelationship com informações de destino e origem.

Exemplo

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Trident criará um SnapMirror relacionamento com o nome da política de relacionamento configurada (ou padrão para ONTAP) e o inicializará.

c. Crie um PVC com o StorageClass previamente criado para atuar como secundário (SnapMirror destino).

Exemplo

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Trident verificará o CRD TridentMirrorRelationship e não conseguirá criar o volume se o relacionamento não existir. Se o relacionamento existir, Trident garantirá que o novo volume FlexVol seja colocado em uma SVM que esteja emparelhada com a SVM remota definida no MirrorRelationship.

Estados de replicação de volume

Um Trident Mirror Relationship (TMR) é um CRD que representa uma das extremidades de uma relação de replicação entre PVCs. O TMR de destino possui um estado, que informa ao Trident qual é o estado desejado. O TMR de destino possui os seguintes estados:

- **Estabelecido**: o PVC local é o volume de destino de uma relação de espelhamento, e esta é uma nova relação.
- **Promovido**: o PVC local é ReadWrite e montável, sem relação de espelhamento em vigor no momento.
- **Reestabelecido**: o PVC local é o volume de destino de uma relação de espelhamento e também estava anteriormente nessa relação de espelhamento.
  - O estado restabelecido deve ser usado se o volume de destino já teve alguma relação com o volume de origem, pois ele sobrescreve o conteúdo do volume de destino.
  - O estado restabelecido falhará se o volume não estava previamente em uma relação com a origem.

Promover o PVC secundário durante uma falha não planejada

Execute a seguinte etapa no cluster Kubernetes secundário:

- Atualize o campo *spec.state* de TridentMirrorRelationship para `promoted`.

Promover o PVC secundário durante uma falha planejada

Durante um failover (migração) planejado, execute as seguintes etapas para promover o PVC secundário:

Passos

1. No cluster Kubernetes primário, crie um snapshot do PVC e aguarde até que o snapshot seja criado.
2. No cluster Kubernetes primário, crie o SnapshotInfo CR para obter detalhes internos.

   Exemplo

   ```
   kind: SnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshot-name: csi-nas-snapshot
   ```

3. No cluster Kubernetes secundário, atualize o campo *spec.state* do CR *TridentMirrorRelationship* para *promoted* e *spec.promotedSnapshotHandle* para ser o internalName do snapshot.
4. No cluster Kubernetes secundário, confirme o status (campo status.state) de TridentMirrorRelationship para promovido.

Restaurar uma relação de espelhamento após uma falha

Antes de restaurar a relação de espelhamento, escolha o lado que deseja definir como o novo primário.

Passos

1. No cluster Kubernetes secundário, verifique se os valores do campo *spec.remoteVolumeHandle* em TridentMirrorRelationship foram atualizados.
2. No cluster Kubernetes secundário, atualize o campo *spec.mirror* de TridentMirrorRelationship para `reestablished`.

Operações adicionais

Trident suporta as seguintes operações nos volumes primário e secundário:

Replicar o PVC primário em um novo PVC secundário

Certifique-se de que você já possui um PVC primário e um PVC secundário.

Passos

1. Exclua os CRDs PersistentVolumeClaim e TridentMirrorRelationship do cluster secundário (destino) estabelecido.
2. Exclua o CRD TridentMirrorRelationship do cluster primário (de origem).
3. Crie um novo TridentMirrorRelationship CRD no cluster primário (origem) para o novo PVC secundário (destino) que você deseja estabelecer.

Redimensione um PVC espelhado, primário ou secundário

O PVC pode ser redimensionado normalmente, o ONTAP expandirá automaticamente qualquer FlexVol volume de destino se a quantidade de dados exceder o tamanho atual.

Remover replicação de um PVC

Para remover a replicação, execute uma das seguintes operações no volume secundário atual:

- Exclua o MirrorRelationship no PVC secundário. Isso interrompe a relação de replicação.
- Ou, atualize o campo spec.state para *promoted*.

Excluir um PVC (que foi previamente espelhado)

Trident verifica se há PVCs replicados e libera a relação de replicação antes de tentar excluir o volume.

Excluir um TMR

A exclusão de um TMR em um dos lados de um relacionamento espelhado faz com que o TMR restante passe para o estado *promovido* antes que o Trident conclua a exclusão. Se o TMR selecionado para exclusão já estiver no estado *promovido*, não haverá um relacionamento espelhado existente e o TMR será removido e o Trident promoverá o PVC local para *ReadWrite*. Essa exclusão libera os metadados SnapMirror do volume local no ONTAP. Se esse volume for usado em um relacionamento espelhado no futuro, será necessário usar um novo TMR com um estado de replicação de volume *estabelecido* ao criar o novo relacionamento espelhado.

Atualize os relacionamentos de espelhamento quando ONTAP estiver online

As relações de espelhamento podem ser atualizadas a qualquer momento após serem estabelecidas. Você pode usar o `state: promoted` ou `state: reestablished` campos para atualizar as relações. Ao promover um volume de destino para um volume ReadWrite regular, você pode usar *promotedSnapshotHandle* para especificar um snapshot específico para restaurar o volume atual.

Atualizar relações de espelhamento quando ONTAP estiver offline

Você pode usar um CRD para realizar uma atualização SnapMirror sem que o Trident tenha conectividade direta com o cluster ONTAP. Consulte o seguinte exemplo de formato do TridentActionMirrorUpdate:

Exemplo

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` reflete o estado do TridentActionMirrorUpdate CRD. Pode assumir um valor de *Succeeded*, *In Progress* ou *Failed*.

Usar a topologia CSI

Trident pode criar e anexar volumes seletivamente aos nós presentes em um cluster Kubernetes fazendo uso do "Recurso de topologia CSI".

Visão geral

Utilizando o recurso de CSI Topology, o acesso a volumes pode ser limitado a um subconjunto de nós, com base em regiões e zonas de disponibilidade. Atualmente, os provedores de nuvem permitem que os administradores do Kubernetes criem nós baseados em zonas. Os nós podem estar localizados em diferentes zonas de disponibilidade dentro de uma região ou em várias regiões. Para facilitar o provisionamento de volumes para cargas de trabalho em uma arquitetura multizona, Trident utiliza CSI Topology.

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

Kubernetes oferece dois modos exclusivos de vinculação de volumes:

- Com `VolumeBindingMode` definido como `Immediate`, Trident cria o volume sem qualquer reconhecimento de topologia. A vinculação de volume e o provisionamento dinâmico são tratados quando o PVC é criado. Este é o padrão `VolumeBindingMode` e é adequado para clusters que não impõem restrições de topologia. 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 são adiadas até que um pod que utiliza o PVC seja agendado e criado. Dessa forma, os volumes são criados para atender às restrições de agendamento impostas pelos requisitos de topologia.

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

O que você vai precisar

Para utilizar a topologia CSI, você precisa do seguinte:

- Um cluster Kubernetes executando um "versão do Kubernetes suportada"

```
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 do cluster devem ter rótulos que introduzam a consciência de topologia (`topology.kubernetes.io/region` e `topology.kubernetes.io/zone`). Esses rótulos **devem estar presentes nos nós do cluster** antes que o Trident seja instalado para que o Trident reconheça a 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 backend com reconhecimento de topologia

Os backends de storage Trident podem ser projetados para provisionar volumes seletivamente com base em zonas de disponibilidade. Cada backend pode conter um bloco `supportedTopologies` opcional que representa uma lista de zonas e regiões suportadas. Para StorageClasses que fazem uso de tal backend, um volume só será criado se solicitado por uma aplicação agendada em uma região/zona suportada.

Aqui está um exemplo de definição de backend:

YAML

```yaml
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-a
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-b
```

JSON

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

OBSERVAÇÃO	`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 StorageClasses que contêm um subconjunto das regiões e zonas fornecidas em um backend, Trident cria um volume no backend.

Você pode definir `supportedTopologies` por pool de storage também. 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: password
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
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-b
```

Neste exemplo, os `region` e `zone` rótulos representam a localização do pool de storage. `topology.kubernetes.io/region` e `topology.kubernetes.io/zone` indicam de onde os pools de storage podem ser consumidos.

Etapa 2: Defina StorageClasses que são cientes da topologia

Com base nos rótulos de topologia fornecidos aos nós do cluster, StorageClasses podem ser definidos para conter informações de topologia. Isso determinará os pools de storage que servem como candidatos para 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 de StorageClass fornecida acima, `volumeBindingMode` está definido como `WaitForFirstConsumer`. Os PVCs solicitados com este StorageClass não serão processados até que sejam referenciados em um pod. E, `allowedTopologies` fornece as zonas e a região a serem usadas. O `netapp-san-us-east1` StorageClass cria PVCs no backend `san-backend-us-east1` definido acima.

Etapa 3: criar e usar um PVC

Com o StorageClass criado e mapeado para um backend, agora você pode criar PVCs.

Veja o exemplo `spec` abaixo:

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

A criação de 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 que Trident crie um volume e o vincule 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 que estão presentes na `us-east1` região e escolher entre qualquer nó que esteja presente nas `us-east1-a` ou `us-east1-b` zonas.

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 preexistentes podem ser atualizados para incluir uma lista de `supportedTopologies` usando `tridentctl backend update`. Isso não afetará os volumes que já foram provisionados e será usado apenas para PVCs subsequentes.

Encontre mais informações

- "Gerenciar recursos para contêineres"
- "nodeSelector"
- "Afinidade e antiafinidade"
- "Taints e Tolerations"

Trabalhar com Snapshots

Os snapshots de volumes persistentes (PVs) do Kubernetes permitem cópias de volumes em ponto no tempo. Você pode criar um snapshot de um volume criado usando Trident, importar um snapshot criado fora do Trident, criar um novo volume a partir de um snapshot existente e recuperar dados de volumes a partir de snapshots.

Visão geral

O snapshot de volume é suportado pelos `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files` e `google-cloud-netapp-volumes` drivers.

Antes de começar

Você precisa de um controlador de snapshots externo e definições de recursos personalizados (CRDs) para trabalhar com snapshots. Essa é a responsabilidade do orquestrador do Kubernetes (por exemplo: Kubeadm, GKE, OpenShift).

Se a sua distribuição Kubernetes não incluir o snapshot controller e os CRDs, consulte Implante um controlador de Snapshot de volume.

OBSERVAÇÃO	Não crie um controlador de snapshot se estiver criando snapshots de volume sob demanda em um ambiente GKE. O GKE usa um controlador de snapshot incorporado e oculto.

Criar um snapshot de volume

Passos

1. Crie um `VolumeSnapshotClass`. Para obter mais informações, consulte "VolumeSnapshotClass".
   - O `driver` aponta para o driver Trident CSI.
   - `deletionPolicy` pode ser `Delete` ou `Retain`. Quando definido como `Retain`, o snapshot físico subjacente no cluster de armazenamento é mantido mesmo quando o objeto `VolumeSnapshot` é excluído.

     Exemplo

     ```
     cat snap-sc.yaml
     ```

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

2. Crie um snapshot de um PVC existente.

   Exemplos

   - Este exemplo cria um snapshot de um PVC existente.

     ```
     cat snap.yaml
     ```

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

   - Este exemplo cria um objeto de snapshot de volume para um PVC nomeado `pvc1` e o nome do snapshot é definido como `pvc1-snap`. Um VolumeSnapshot é análogo a um PVC e está associado a um `VolumeSnapshotContent` objeto que representa o snapshot real.

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

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

- Você pode identificar o `VolumeSnapshotContent` objeto para o `pvc1-snap` VolumeSnapshot descrevendo-o. O `Snapshot Content Name` identifica o objeto VolumeSnapshotContent que serve a este snapshot. O `Ready To Use` parâmetro indica que o snapshot pode ser usado para criar um novo PVC.

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

Crie um PVC a partir de um instantâneo de volume

Você pode usar `dataSource` para criar um PVC usando um VolumeSnapshot chamado `<pvc-name>` como fonte dos dados. Depois que o PVC é criado, ele pode ser anexado a um pod e usado como qualquer outro PVC.

AVISO	O PVC será criado no mesmo backend que o volume de origem. Consulte "KB: A criação de um PVC a partir de um Trident PVC Snapshot não pode ser feita em um backend alternativo".

O exemplo a seguir cria o PVC usando `pvc1-snap` como fonte de dados.

```
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
```

Importar um Snapshot de volume

Trident oferece suporte à "Processo de snapshot pré-provisionado do Kubernetes" para permitir que o administrador do cluster crie um `VolumeSnapshotContent` objeto e importe snapshots criados fora do Trident.

Antes de começar

Trident deve ter criado ou importado o volume pai do Snapshot.

Passos

1. **Administrador do cluster:** Crie um `VolumeSnapshotContent` objeto que faça referência ao snapshot do backend. Isso inicia o fluxo de trabalho de snapshot no Trident.
   - Especifique o nome do snapshot do backend em `annotations` como `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
   - Especifique `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` em `snapshotHandle`. Esta é a única informação fornecida ao Trident pelo snapshotter externo na chamada `ListSnapshots`.

OBSERVAÇÃO	O `<volumeSnapshotContentName>` não pode sempre corresponder ao nome do snapshot do backend devido a restrições de nomenclatura do CR.

     Exemplo

     O exemplo a seguir cria um `VolumeSnapshotContent` objeto que faz referência ao snapshot do backend `snap-01`.

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

2. **Administrador do cluster:** Crie o `VolumeSnapshot` CR que referencia o `VolumeSnapshotContent` objeto. Isso solicita acesso para usar o `VolumeSnapshot` em um determinado namespace.

   Exemplo

   O exemplo a seguir cria um `VolumeSnapshot` CR chamado `import-snap` que faz referência ao `VolumeSnapshotContent` chamado `import-snap-content`.

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

3. **Processamento interno (nenhuma ação necessária):** O snapshotter externo reconhece o recém-criado `VolumeSnapshotContent` e executa a `ListSnapshots` chamada. Trident cria o `TridentSnapshot`.
   - O snapshotter externo define o `VolumeSnapshotContent` para `readyToUse` e o `VolumeSnapshot` para `true`.
   - Trident está de volta `readyToUse=true`.
4. **Qualquer usuário:** Crie um `PersistentVolumeClaim` para referenciar o novo `VolumeSnapshot`, onde o `spec.dataSource` (ou `spec.dataSourceRef`) nome é o nome `VolumeSnapshot`.

   Exemplo

O exemplo a seguir cria um PVC referenciando o `VolumeSnapshot` nomeado `import-snap`.

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

Recupere dados de volume usando Snapshots

O diretório de snapshots está oculto por padrão para facilitar a máxima compatibilidade de volumes provisionados usando os `ontap-nas` e `ontap-nas-economy` drivers. Habilite o `.snapshot` diretório para recuperar dados diretamente dos snapshots.

Use o volume snapshot restore ONTAP CLI para restaurar um volume a um estado registrado em um snapshot anterior.

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

OBSERVAÇÃO	Ao restaurar uma cópia de Snapshot, a configuração do volume existente é sobrescrita. As alterações feitas nos dados do volume após a criação da cópia de Snapshot são perdidas.

Restauração de volume in-place a partir de uma Snapshot

Trident oferece restauração rápida e in-place de volumes a partir de um snapshot usando o `TridentActionSnapshotRestore` (TASR) CR. Este CR funciona como uma ação imperativa do Kubernetes e não persiste após a conclusão da operação.

Trident suporta a restauração de snapshot nos `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes` e `solidfire-san` drivers.

Antes de começar

Você deve ter um PVC vinculado e um snapshot de volume disponível.

- Verifique se o status do PVC está bound.

```
kubectl get pvc
```

- Verifique se o snapshot do volume está pronto para uso.

```
kubectl get vs
```

Passos

1. Crie o CR TASR. Este exemplo cria um CR para PVC `pvc1` e volume snapshot `pvc1-snapshot`.

   OBSERVAÇÃO O TASR CR deve estar em um namespace onde o PVC e o VS existam.

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. Aplique a CR para restaurar a partir do snapshot. Este exemplo restaura a partir do snapshot `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

Resultados

Trident restaura os dados a partir do snapshot. Você pode verificar o status da restauração do snapshot:

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

OBSERVAÇÃO

- Na maioria dos casos, Trident não tentará novamente a operação automaticamente em caso de falha. Você precisará executar a operação novamente.
- Usuários do Kubernetes sem acesso de administrador podem precisar receber permissão do administrador para criar um TASR CR em seu namespace de aplicação.

Excluir um PV com snapshots associados

Ao excluir um Persistent Volume com snapshots associados, o volume Trident correspondente é atualizado para o estado "Deleting state". Remova os snapshots do volume para excluir o volume Trident.

Implante um controlador de Snapshot de volume

Se a sua distribuição Kubernetes não incluir o snapshot controller e os CRDs, você pode implantá-los da seguinte forma.

Passos

1. Criar CRDs de Snapshot de volume.

```
cat snapshot-setup.sh
```

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

2. Crie o controlador de snapshot.

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

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

OBSERVAÇÃO	Se necessário, abra `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e atualize `namespace` para o seu namespace.

Links relacionados

- "Instantâneos de volume"
- "VolumeSnapshotClass"

Trabalhar com Snapshots de grupo de volume

Snapshots de grupo de volumes do Kubernetes de Persistent Volumes (PVs) NetApp Trident fornecem a capacidade de criar snapshots de múltiplos volumes (um grupo de snapshots de volumes). Esse snapshot de grupo de volumes representa cópias de múltiplos volumes feitas no mesmo ponto no tempo.

OBSERVAÇÃO	VolumeGroupSnapshot é um recurso beta no Kubernetes com APIs beta. Kubernetes 1.32 é a versão mínima necessária para VolumeGroupSnapshot.

Criar snapshots de grupo de volume

O snapshot de grupo de volume é compatível com os seguintes drivers de armazenamento:

- `ontap-san`driver - somente para os protocolos iSCSI e FC, não para o protocolo NVMe/TCP.
- `ontap-san-economy` - somente para o protocolo iSCSI.
- `ontap-nas`

OBSERVAÇÃO	O snapshot de grupo de volume não é compatível com sistemas de armazenamento NetApp ASA r2 ou AFX.

Antes de começar

- Certifique-se de que sua versão do Kubernetes seja K8s 1.32 ou superior.
- Você precisa de um controlador de snapshots externo e definições de recursos personalizados (CRDs) para trabalhar com snapshots. Essa é a responsabilidade do orquestrador do Kubernetes (por exemplo: Kubeadm, GKE, OpenShift).

  Se a sua distribuição Kubernetes não incluir o controlador de snapshot externo e os CRDs, consulte Implante um controlador de Snapshot de volume.

OBSERVAÇÃO	Não crie um controlador de snapshot se estiver criando snapshots de grupo de volume sob demanda em um ambiente GKE. O GKE usa um controlador de snapshot incorporado e oculto.

- No arquivo YAML do controlador de snapshot, defina o `CSIVolumeGroupSnapshot`feature gate como 'true' para garantir que o snapshot de grupo de volume esteja habilitado.
- Crie as classes de snapshot de grupo de volume necessárias antes de criar um snapshot de grupo de volume.
- Certifique-se de que todos os PVCs/volumes estejam no mesmo SVM para poder criar VolumeGroupSnapshot.

Passos

- Crie um VolumeGroupSnapshotClass antes de criar um VolumeGroupSnapshot. Para obter mais informações, consulte "VolumeGroupSnapshotClass".

  ```
  apiVersion: groupsnapshot.storage.k8s.io/v1beta1
  kind: VolumeGroupSnapshotClass
  metadata:
    name: csi-group-snap-class
    annotations:
      kubernetes.io/description: "Trident group snapshot class"
  driver: csi.trident.netapp.io
  deletionPolicy: Delete
  ```

- Crie PVCs com os rótulos necessários usando as classes de armazenamento existentes ou adicione esses rótulos aos PVCs existentes.

  O exemplo a seguir cria o PVC usando `pvc1-group-snap` como fonte de dados e o rótulo

`consistentGroupSnapshot: groupA`. Defina a chave e o valor do rótulo de acordo com suas necessidades.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- Crie um VolumeGroupSnapshot com o mesmo rótulo (`consistentGroupSnapshot: groupA` especificado no PVC.

  Este exemplo cria um Snapshot de grupo de volume:

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

Recuperar dados de volume usando um snapshot de grupo

Você pode restaurar Volumes Persistentes individuais usando os snapshots individuais que foram criados como parte do Snapshot do Grupo de Volumes. Você não pode recuperar o Snapshot do Grupo de Volumes como uma unidade.

Use o volume snapshot restore ONTAP CLI para restaurar um volume a um estado registrado em um snapshot anterior.

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

OBSERVAÇÃO	Ao restaurar uma cópia de Snapshot, a configuração do volume existente é sobrescrita. As alterações feitas nos dados do volume após a criação da cópia de Snapshot são perdidas.

Restauração de volume in-place a partir de uma Snapshot

Trident oferece restauração rápida e in-place de volumes a partir de um snapshot usando o `TridentActionSnapshotRestore` (TASR) CR. Este CR funciona como uma ação imperativa do Kubernetes e não persiste após a conclusão da operação.

Para obter mais informações, consulte "Restauração de volume in-place a partir de uma Snapshot".

Excluir um PV com snapshots de grupo associados

Ao excluir um Snapshot de grupo de volume:

- Você pode excluir VolumeGroupSnapshots como um todo, não snapshots individuais do grupo.
- Se PersistentVolumes forem excluídos enquanto existir um snapshot para esse PersistentVolume, Trident moverá esse volume para um estado de "exclusão", pois o snapshot precisa ser removido antes que o volume possa ser removido com segurança.
- Se um clone tiver sido criado usando um snapshot agrupado e, em seguida, o grupo for excluído, uma operação de divisão no clone será iniciada e o grupo não poderá ser excluído até que a divisão seja concluída.

Implante um controlador de Snapshot de volume

Se a sua distribuição Kubernetes não incluir o snapshot controller e os CRDs, você pode implantá-los da seguinte forma.

Passos

1. Criar CRDs de Snapshot de volume.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. Crie o controlador de snapshot.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

OBSERVAÇÃO Se necessário, abra `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e atualize `namespace` para o seu namespace.

Links relacionados

- "VolumeGroupSnapshotClass"
- "Instantâneos de volume"

Informações sobre direitos autorais

Informações sobre marcas comerciais