



Use o Astra Trident

Astra Trident

NetApp
January 14, 2026

Índice

Use o Astra Trident	1
Prepare o nó de trabalho	1
Selecionar as ferramentas certas	1
Detecção de serviço de nós	1
Volumes NFS	2
Volumes iSCSI	2
Volumes NVMe/TCP	6
Configurar e gerenciar backends	7
Configurar backends	7
Azure NetApp Files	7
Google Cloud NetApp volumes	25
Configure um back-end do Cloud Volumes Service para o Google Cloud	37
Configurar um back-end NetApp HCI ou SolidFire	48
Controladores SAN ONTAP	54
Drivers nas ONTAP	78
Amazon FSX para NetApp ONTAP	106
Crie backends com kubectl	138
Gerenciar backends	145
Criar e gerenciar classes de armazenamento	154
Crie uma classe de armazenamento	154
Gerenciar classes de armazenamento	157
Provisionar e gerenciar volumes	159
Provisionar um volume	159
Expanda volumes	164
Importar volumes	171
Personalizar nomes e rótulos de volume	178
Compartilhar um volume NFS entre namespaces	181
Replique volumes usando o SnapMirror	185
Use a topologia CSI	201
Trabalhar com instantâneos	209

Use o Astra Trident

Prepare o nó de trabalho

Todos os nós de trabalho no cluster do Kubernetes precisam ser capazes de montar os volumes provisionados para os pods. Para preparar os nós de trabalho, é necessário instalar ferramentas NFS, iSCSI ou NVMe/TCP com base na seleção de driver.

Selecionar as ferramentas certas

Se você estiver usando uma combinação de drivers, você deve instalar todas as ferramentas necessárias para seus drivers. Versões recentes do RedHat CoreOS têm as ferramentas instaladas por padrão.

Ferramentas NFS

"[Instalar as ferramentas NFS](#)" se estiver a utilizar: `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `azure-netapp-files` `gcp-cvs`.

Ferramentas iSCSI

"[Instale as ferramentas iSCSI](#)" se estiver a utilizar: `ontap-san`, `ontap-san-economy`, `solidfire-san`.

Ferramentas NVMe

"[Instalar as ferramentas do NVMe](#)" Se você estiver usando `ontap-san` o protocolo NVMe (não-volátil Memory Express) em TCP (NVMe/TCP).



Recomendamos o ONTAP 9.12 ou posterior para NVMe/TCP.

Detecção de serviço de nós

O Astra Trident tenta detetar automaticamente se o nó pode executar serviços iSCSI ou NFS.



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

Rever eventos

O Astra Trident cria eventos para o nó a fim de identificar os serviços descobertos. Para rever estes eventos, execute:

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

Reveja os serviços descobertos

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

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

Volumes NFS

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

RHEL 8 MAIS

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

Ubuntu

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



Reinicie seus nós de trabalho após instalar as ferramentas NFS para evitar falhas ao anexar volumes a contêineres.

Volumes iSCSI

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

Recursos de autorrecuperação iSCSI

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

1. **Identifique** o estado de sessão iSCSI desejado e o estado atual da sessão iSCSI.
2. **Compare** o estado desejado com o estado atual para identificar as reparações necessárias. O Astra Trident determina as prioridades de reparação e quando efetuar as reparações.
3. **Efetuar reparações** necessárias para repor o estado atual da sessão iSCSI para o estado de sessão iSCSI pretendido.



Logs de atividade de auto-cura estão localizados no `trident-main` recipiente no respectivo pod Daemonset. Para visualizar logs, você deve ter definido `debug` como `"true"` durante a instalação do Astra Trident.

As funcionalidades de autorrecuperação iSCSI do Astra Trident ajudam a impedir:

- Sessões iSCSI obsoletas ou não saudáveis que podem ocorrer após um problema de conectividade de rede. No caso de uma sessão obsoleta, o Astra Trident aguarda sete minutos antes de sair para restabelecer a conexão com um portal.



Por exemplo, se os segredos CHAP foram girados no controlador de armazenamento e a rede perder a conectividade, os segredos CHAP antigos (*stale*) podem persistir. A auto-cura pode reconhecer isso e restabelecer automaticamente a sessão para aplicar os segredos CHAP atualizados.

- Sessões iSCSI em falta
- LUNs em falta

Pontos a considerar antes de atualizar o Trident

- Se apenas os grupos por nó (introduzidos em mais de 23,04) estiverem em uso, a recuperação automática iSCSI iniciará os rescans SCSI para todos os dispositivos no barramento SCSI.
- Se apenas grupos com escopo de back-end (obsoletos a partir de 23,04) estiverem em uso, a recuperação automática iSCSI iniciará as reconfigurações SCSI para IDs de LUN exatas no barramento SCSI.
- Se uma combinação de grupos por nó e grupos com escopo de back-end estiver em uso, a recuperação automática iSCSI iniciará as substituições SCSI para IDs LUN exatas no barramento SCSI.

Instale as ferramentas iSCSI

Instale as ferramentas iSCSI utilizando os comandos do seu sistema operativo.

Antes de começar

- Cada nó no cluster do Kubernetes precisa ter uma IQN exclusiva. **Este é um pré-requisito necessário.**
- Se estiver usando RHCOS versão 4,5 ou posterior, ou outra distribuição Linux compatível com RHEL, com o `solidfire-san` driver e o Element OS 12,5 ou anterior, verifique se o algoritmo de autenticação CHAP está definido como MD5 em `/etc/iscsi/iscsid.conf`. algoritmos CHAP compatíveis com FIPS seguros SHA1, SHA-256 e SHA3-256 estão disponíveis com o elemento 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/RedHat CoreOS com iSCSI PVs, especifique a `discard mountOption` no `StorageClass` para executar a recuperação de espaço em linha. Consulte a ["Documentação da RedHat"](#).

RHEL 8 MAIS

1. Instale os seguintes pacotes de sistema:

```
sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-mapper-multipath
```

2. Verifique se a versão iscsi-iniciador-utils é 6,2.0,874-2.el7 ou posterior:

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

3. Definir a digitalização para manual:

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

4. Ativar multipathing:

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



Certifique-se de `etc/multipath.conf` que contém `find_multipaths` no `defaults` em .

5. Certifique-se de que `iscsid` e `multipathd` estão a funcionar:

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

6. Ativar e iniciar `iscsi`:

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

Ubuntu

1. Instale os seguintes pacotes de sistema:

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

2. Verifique se a versão Open-iscsi é 2,0.874-5ubuntu2.10 ou posterior (para bionic) ou 2,0.874-7.1ubuntu6.1 ou posterior (para focal):

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

3. Definir a digitalização para manual:

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

4. Ativar 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 `etc/multipath.conf` que contém `find_multipaths no` em `defaults` em .

5. Certifique-se de que `open-iscsi` e `multipath-tools` estão ativados 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 portas de destino com `iscsiadm` antes de iniciar `open-iscsi` o daemon iSCSI para iniciar. Em alternativa, pode modificar o `iscsi` serviço para iniciar `iscsid` automaticamente.

Configurar ou desativar a auto-recuperação iSCSI

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

- **Intervalo de auto-recuperação iSCSI:** Determina a frequência na qual a auto-recuperação iSCSI é invocada (predefinição: 5 minutos). Você pode configurá-lo para executar com mais frequência definindo um número menor ou com menos frequência definindo um número maior.



Definir o intervalo de auto-recuperação iSCSI para 0 interrompe completamente a auto-recuperação iSCSI. Não recomendamos a desativação do iSCSI Self-healing; ele só deve ser desativado em certos cenários quando o iSCSI Self-healing não estiver funcionando como pretendido ou para fins de depuração.

- **Tempo de espera de auto-cura iSCSI:** Determina a duração de espera de auto-recuperação iSCSI antes de sair de uma sessão não saudável e tentar iniciar sessão novamente (predefinição: 7 minutos). Você pode configurá-lo para um número maior para que as sessões identificadas como não saudáveis tenham que esperar mais antes de serem desconetadas e, em seguida, uma tentativa é feita para fazer login novamente, ou um número menor para fazer logout e fazer login mais cedo.

Leme

Para configurar ou alterar as definições de recuperação automática iSCSI, passe os `iscsiSelfHealingInterval` parâmetros e `iscsiSelfHealingWaitTime` durante a instalação do leme ou atualização do leme.

O exemplo a seguir define o intervalo de auto-recuperação iSCSI para 3 minutos e o tempo de espera de auto-recuperação para 6 minutos:

```
helm install trident trident-operator-100.2406.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

tridentctl

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

O exemplo a seguir define o intervalo de auto-recuperação iSCSI para 3 minutos e o tempo de espera de auto-recuperação 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 do seu sistema operacional.



- O NVMe requer o 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 versão do kernel, talvez seja necessário atualizar a versão do kernel do 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
```

Verifique a instalação

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

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



O Astra Trident modifica o `ctrl_device_tmo` valor para garantir que o NVMe não desista do caminho se ele cair. Não altere esta definição.

Configurar e gerenciar backends

Configurar backends

Um back-end define a relação entre o Astra Trident e um sistema de storage. Ele diz ao Astra Trident como se comunicar com esse sistema de storage e como o Astra Trident deve provisionar volumes a partir dele.

O Astra Trident oferece automaticamente pools de storage de back-ends que atendem aos requisitos definidos por uma classe de storage. Saiba como configurar o back-end para o seu sistema de armazenamento.

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

Azure NetApp Files

Configurar um back-end do Azure NetApp Files

Você pode configurar o Azure NetApp Files como back-end para o Astra Trident. É possível anexar volumes NFS e SMB usando um back-end do Azure NetApp Files. O Astra Trident também oferece suporte ao gerenciamento de credenciais usando identidades gerenciadas para clusters do Azure Kubernetes Services (AKS).

Detalhes do driver Azure NetApp Files

O Astra Trident fornece os seguintes drivers de storage Azure NetApp Files para se comunicar com o cluster. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Condutor	Protocolo	VolumeMo de	Modos de acesso suportados	Sistemas de arquivos suportados
azure-netapp-files	NFS, SMB	Sistema de ficheiros	RWO, ROX, RWX, RWOP	nfs, smb

Considerações

- O serviço Azure NetApp Files não oferece suporte a volumes menores que 100 GB. O Astra Trident cria automaticamente volumes de 100 GiB se um volume menor for solicitado.
- O Astra Trident é compatível com volumes SMB montados em pods executados apenas em nós do Windows.

Identities gerenciadas para AKS

O Astra Trident é compatível "[identidades gerenciadas](#)" com clusters do Azure Kubernetes Services. Para aproveitar o gerenciamento simplificado de credenciais oferecido por identidades gerenciadas, você deve ter:

- Um cluster do Kubernetes implantado usando AKS
- Identidades gerenciadas configuradas no cluster AKS kuquilla
- Astra Trident instalado que inclui o `cloudProvider` para especificar "Azure".

Operador Trident

Para instalar o Astra Trident usando o operador Trident, edite `tridentorchestrator_cr.yaml` para definir `cloudProvider` como "Azure". Por exemplo:

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

Leme

O exemplo a seguir instala conjuntos Astra Trident `cloudProvider` no Azure usando a variável de ambiente `$CP`:

```
helm install trident trident-operator-100.2406.0.tgz --create
--namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

`tridentctl`

O exemplo a seguir instala o Astra Trident e define o `cloudProvider` sinalizador como Azure:

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

Identidade de nuvem para AKS

A identidade na nuvem permite que os pods do Kubernetes acessem recursos do Azure autenticando como uma identidade de workload em vez de fornecendo credenciais explícitas do Azure.

Para aproveitar a identidade da nuvem no Azure, você deve ter:

- Um cluster do Kubernetes implantado usando AKS
- Identidade da carga de trabalho e `oidc-emitente` configurados no cluster AKS Kubernetes
- Astra Trident instalado que inclui o `cloudProvider` para especificar "Azure" e `cloudIdentity` especificar a identidade da carga de trabalho

Operador Trident

Para instalar o Astra Trident usando o operador Trident, edite `tridentorchestrator_cr.yaml` para definir `cloudProvider` como "Azure" e defina `cloudIdentity` como `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.

Por exemplo:

```
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-xxxxxxxxxxxx' *
```

Leme

Defina os valores para sinalizadores **provedor de nuvem (CP)** e **identidade de nuvem (IC)** usando as seguintes variáveis de ambiente:

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

O exemplo a seguir instala o Astra Trident e define `cloudProvider` para o Azure usando a variável de ambiente `$CP` e define a `cloudIdentity` variável usando o ambiente `$CI`:

```
helm install trident trident-operator-100.2406.0.tgz --set
cloudProvider=$CP --set cloudIdentity=$CI
```

<code>dtridentctl</code>

Defina os valores para os sinalizadores **provedor de nuvem** e **identidade de nuvem** usando as seguintes variáveis de ambiente:

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

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

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

Prepare-se para configurar um back-end do Azure NetApp Files

Antes de configurar o back-end do Azure NetApp Files, você precisa garantir que os seguintes requisitos sejam atendidos.

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 a ["Azure: Configure o Azure NetApp Files e crie um volume NFS"](#).

Para configurar e usar um ["Azure NetApp Files"](#) back-end, você precisa do seguinte:



- `subscriptionID` `tenantID`, `clientID`, `location` E `clientSecret` são opcionais ao usar identidades gerenciadas em um cluster AKS.
- `tenantID` `clientID`, `clientSecret` são opcionais ao usar uma identidade de nuvem em um cluster AKS.

- Um pool de capacidade. ["Microsoft: Crie um pool de capacidade para o Azure NetApp Files"](#)Consulte a .
- Uma sub-rede delegada ao Azure NetApp Files. ["Microsoft: Delegar uma sub-rede ao Azure NetApp Files"](#)Consulte a .
- `subscriptionID` A partir de uma subscrição do Azure com o Azure NetApp Files ativado.
- `tenantID`, `clientID` E `clientSecret` de um ["Registro da aplicação"](#) no Azure active Directory com permissões suficientes para o serviço Azure NetApp Files. O Registro de aplicativos deve usar:
 - A função proprietário ou Colaborador ["Pré-definido pelo Azure"](#).
 - A ["Função de Colaborador personalizada"](#)no nível da subscrição (`assignableScopes`) com as seguintes permissões limitadas apenas ao que o Astra Trident requer. Depois de criar a função personalizada ["Atribua a função usando o portal do Azure"](#), .

Função de colaborador personalizada

```
{
  "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/MountTargets/read",

          "Microsoft.Network/virtualNetworks/read",

"Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrations/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 de back-end. Os valores de localização especificados em pools virtuais são ignorados.
- Para usar Cloud Identity`o , obtenha o `client ID de a ["identidade gerenciada atribuída pelo usuário"](#) e especifique esse ID no azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

Requisitos adicionais para volumes SMB

Para criar um volume SMB, você deve ter:

- Active Directory configurado e conectado ao Azure NetApp Files. ["Microsoft: Crie e gerencie conexões do active Directory para Azure NetApp Files"](#) Consulte a .
- Um cluster do Kubernetes com um nó de controlador Linux e pelo menos um nó de trabalho do Windows que executa o Windows Server 2022. O Astra Trident é compatível com volumes SMB montados em pods executados apenas em nós do Windows.
- Pelo menos um segredo do Astra Trident que contém suas credenciais do active Directory para que o Azure NetApp Files possa se autenticar no active Directory. Para gerar segredo smbcreds:

```

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

```

- Um proxy CSI configurado como um serviço Windows. Para configurar um csi-proxy, ["GitHub: CSI"](#)

[Proxy](#)" consulte ou ["GitHub: CSI Proxy para Windows"](#) para nós do Kubernetes executados no Windows.

Exemplos e opções de configuração de back-end do Azure NetApp Files

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

Opções de configuração de back-end

O Astra Trident usa sua configuração de back-end (sub-rede, rede virtual, nível de serviço e local) para criar volumes Azure NetApp Files em pools de capacidade disponíveis no local solicitado e que correspondam ao nível de serviço e à sub-rede solicitados.



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

Os backends Azure NetApp Files fornecem essas opções de configuração.

Parâmetro	Descrição	Padrão
version		Sempre 1
storageDriverName	Nome do controlador de armazenamento	"ficheiros azure-NetApp"
backendName	Nome personalizado ou back-end de storage	Nome do condutor e caracteres aleatórios
subscriptionID	O ID de assinatura da sua assinatura do Azure Opcional quando identidades gerenciadas está habilitado em um cluster AKS.	
tenantID	O ID do locatário de um Registo de aplicações Opcional quando identidades geridas ou identidade na nuvem são utilizadas num cluster AKS.	
clientID	A ID do cliente de um registo de aplicações opcional quando identidades geridas ou identidade na nuvem são utilizadas num cluster AKS.	
clientSecret	O segredo do cliente de um Registo de aplicações Opcional quando identidades geridas ou identidade na nuvem são utilizadas num cluster AKS.	
serviceLevel	Um de Standard, Premium, ou Ultra	"" (aleatório)

Parâmetro	Descrição	Padrão
location	Nome do local do Azure onde os novos volumes serão criados Opcional quando identidades gerenciadas estiverem ativadas em um cluster AKS.	
resourceGroups	Lista de grupos de recursos para filtragem de recursos descobertos	[] (sem filtro)
netappAccounts	Lista de contas do NetApp para filtragem de recursos descobertos	[] (sem filtro)
capacityPools	Lista de pools de capacidade para filtrar recursos descobertos	[] (sem filtro, aleatório)
virtualNetwork	Nome de uma rede virtual com uma sub-rede delegada	""
subnet	Nome de uma sub-rede delegada Microsoft.Netapp/volumes	""
networkFeatures	Conjunto de recursos VNet para um volume, pode ser Basic ou Standard. Os recursos de rede não estão disponíveis em todas as regiões e podem ter que ser ativados em uma assinatura. Especificar networkFeatures quando a funcionalidade não está ativada faz com que o provisionamento de volume falhe.	""
nfsMountOptions	Controle refinado das opções de montagem NFS. Ignorado para volumes SMB. Para montar volumes usando o NFS versão 4,1, inclua `nfsvers=4` na lista de opções de montagem delimitadas por vírgulas 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 de back-end.	"3"
limitVolumeSize	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor	"" (não aplicado por padrão)

Parâmetro	Descrição	Padrão
debugTraceFlags	Debug flags para usar ao solucionar problemas. Exemplo, <code>\{"api": false, "method": true, "discovery": true\}</code> . Não use isso a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	nulo
nasType	Configurar a criação de volumes NFS ou SMB. As opções são <code>nfs</code> , <code>smb</code> ou <code>null</code> . A configuração como <code>null</code> padrão para volumes NFS.	<code>nfs</code>
supportedTopologies	Representa uma lista de regiões e zonas que são suportadas por este backend. Para obter mais informações, " Use a topologia CSI " consulte .	



Para obter mais informações sobre recursos de rede, "[Configurar recursos de rede para um volume Azure NetApp Files](#)" consulte .

Permissões e recursos necessários

Se você receber um erro "sem pools de capacidade encontrados" ao criar um PVC, é provável que o Registro do aplicativo não tenha as permissões e recursos necessários (sub-rede, rede virtual, pool de capacidade) associados. Se a depuração estiver ativada, o Astra Trident registrará os recursos do Azure descobertos quando o back-end for 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 vários recursos com o mesmo nome.

Os `resourceGroups` valores, `netappAccounts`, e `capacityPools` são filtros que restringem o conjunto de recursos descobertos aos disponíveis para esse back-end de armazenamento e podem ser especificados em qualquer combinação. Nomes totalmente qualificados seguem este formato:

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

Provisionamento de volume

Você pode controlar o provisionamento de volume padrão especificando as seguintes opções em uma seção especial do arquivo de configuração. [Exemplos de configurações](#) Consulte para obter detalhes.

Parâmetro	Descrição	Padrão
<code>exportRule</code>	Regras de exportação para novos volumes. <code>exportRule</code> 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"
<code>snapshotDir</code>	Controla a visibilidade do diretório <code>.snapshot</code>	"falso"
<code>size</code>	O tamanho padrão dos novos volumes	"100G"
<code>unixPermissions</code>	As permissões unix de novos volumes (4 dígitos octal). Ignorado para volumes SMB.	"" (recurso de pré-visualização, requer lista branca na assinatura)

Exemplos de configurações

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

Configuração mínima

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

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

```
---
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
```

Identities gerenciadas para AKS

Esta configuração de back-end omits `subscriptionID`, `tenantID`, `clientID`, e `clientSecret`, que são opcionais ao usar identities gerenciadas.

```
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
```

Identidade de nuvem para AKS

Essa configuração de back-end omits , tenantID clientID, e clientSecret, que são opcionais ao usar uma identidade de 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 back-end coloca volumes no local do Azure eastus em um Ultra pool de capacidade. O Astra Trident descobre automaticamente todas as sub-redes delegadas 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
```

Configuração avançada

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

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
virtualNetwork: my-virtual-network
subnet: my-subnet
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 do pool virtual

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

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

Configuração de topologias compatíveis

O Astra Trident facilita o provisionamento de volumes para workloads com base em regiões e zonas de disponibilidade. O `supportedTopologies` bloco nesta configuração de back-end é usado para fornecer uma lista de regiões e zonas por back-end. Os valores de região e zona especificados aqui devem corresponder aos valores de região e zona dos rótulos em cada nó de cluster do 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 back-end, o Astra Trident criará volumes na região e na zona mencionadas. Para obter mais informações, ["Use a topologia CSI"](#) consulte .

```
---
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 definições a seguir `StorageClass` referem-se aos pools de armazenamento acima.

Exemplos de definições usando `parameter.selector` campo

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


```

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

```

Definições de exemplo 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 diferentes segredos 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}
```

Usando diferentes segredos por 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 compatíveis com volumes SMB. `nasType: nfs` Ou `nasType: null` filtros para NFS Pools.

Crie o backend

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

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

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

```
tridentctl logs
```

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

Google Cloud NetApp volumes

Configurar um back-end do Google Cloud NetApp volumes

Agora você pode configurar o Google Cloud NetApp volumes como o back-end para o Astra Trident. É possível anexar volumes NFS usando um back-end do Google Cloud NetApp volumes.

```
Google Cloud NetApp Volumes is a tech preview feature in Astra Trident 24.06.
```

Detalhes do driver do Google Cloud NetApp volumes

O Astra Trident fornece ao `google-cloud-netapp-volumes` condutor a comunicação com o cluster. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Condutor	Protocolo	VolumeMo de	Modos de acesso suportados	Sistemas de arquivos suportados
<code>google-cloud-netapp-volumes</code>	NFS	Sistema de arquivos	RWO, ROX, RWX, RWOP	<code>nfs</code>

Prepare-se para configurar um back-end do Google Cloud NetApp volumes

Antes de configurar o back-end do Google Cloud NetApp volumes, você precisa garantir que os requisitos a seguir sejam atendidos.

Pré-requisitos para volumes NFS

Se você estiver usando o Google Cloud NetApp volumes pela primeira vez ou em um novo local, precisará de alguma configuração inicial para configurar o Google Cloud NetApp volumes e criar um volume NFS. ["Antes de começar"](#) Consulte a .

Antes de configurar o back-end do Google Cloud NetApp volumes:

- Uma conta do Google Cloud configurada com o serviço Google Cloud NetApp volumes. ["Google Cloud NetApp volumes"](#) Consulte a .
- Número do projeto da sua conta do Google Cloud. ["Identificação de projetos"](#) Consulte a .
- Uma conta de serviço do Google Cloud com a (`netappcloudvolumes.admin` função de administrador do NetApp volumes). ["Funções e permissões de gerenciamento de identidade e acesso"](#) Consulte a .
- Arquivo de chave de API para sua conta GCNV. Consulte ["Autentique usando chaves de API"](#)
- Um pool de armazenamento. ["Visão geral dos pools de armazenamento"](#) Consulte a .

Para obter mais informações sobre como configurar o acesso ao Google Cloud NetApp volumes, ["Configurar o acesso ao Google Cloud NetApp volumes"](#) consulte .

Exemplos e opções de configuração de back-end do Google Cloud NetApp volumes

Saiba mais sobre as opções de configuração de back-end do NFS para o Google Cloud NetApp volumes e revise exemplos de configuração.

Opções de configuração de back-end

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

Parâmetro	Descrição	Padrão
<code>version</code>		Sempre 1
<code>storageDriverName</code>	Nome do controlador de armazenamento	O valor de <code>storageDriverName</code> deve ser especificado como "google-cloud-NetApp-volumes".
<code>backendName</code>	(Opcional) Nome personalizado do back-end de armazenamento	Nome do driver e parte da chave da API
<code>storagePools</code>	Parâmetro opcional usado para especificar pools de armazenamento para criação de volume.	
<code>projectNumber</code>	Número do projeto da conta Google Cloud. O valor é encontrado na página inicial do portal do Google Cloud.	
<code>location</code>	No Google Cloud, o Astra Trident cria volumes GCNV. Ao criar clusters de Kubernetes entre regiões, os volumes criados em a <code>location</code> podem ser usados em workloads programados em nós em várias regiões do Google Cloud. O tráfego entre regiões incorre em um custo adicional.	

Parâmetro	Descrição	Padrão
apiKey	Chave de API para a conta de serviço do Google Cloud com a <code>netappcloudvolumes.admin</code> função. Ele inclui o conteúdo formatado em JSON do arquivo de chave privada de uma conta de serviço do Google Cloud (copiado literalmente no arquivo de configuração de back-end). O <code>apiKey</code> deve incluir pares de chave-valor para as seguintes chaves: <code>type</code> , <code>project_id</code> , <code>client_email</code> , <code>client_id</code> , <code>auth_uri</code> , <code>token_uri</code> , <code>auth_provider_x509_cert_url</code> , e <code>client_x509_cert_url</code> .	
nfsMountOptions	Controle refinado das opções de montagem NFS.	"3"
limitVolumeSize	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor.	"" (não aplicado por padrão)
serviceLevel	O nível de serviço de um pool de storage e seus volumes. Os valores são <code>flex</code> , <code>standard</code> , <code>premium</code> , <code>extreme</code> ou <code>.</code>	
network	Rede do Google Cloud usada para volumes GCNV.	
debugTraceFlags	Debug flags para usar ao solucionar problemas. Exemplo, <code>{"api":false, "method":true}</code> . Não use isso a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	nulo
supportedTopologies	Representa uma lista de regiões e zonas que são suportadas por este backend. Para obter mais informações, "Use a topologia CSI" consulte . Por exemplo: <pre>supportedTopologies: - topology.kubernetes.io/region: europe-west6 topology.kubernetes.io/zone: europe-west6-b</pre>	

Opções de provisionamento de volume

Você pode controlar o provisionamento de volume padrão `defaults` na seção 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 <code>.snapshot</code> diretório	"falso"
snapshotReserve	Porcentagem de volume reservado para snapshots	"" (aceitar predefinição de 0)

Parâmetro	Descrição	Padrão
<code>unixPermissions</code>	As permissões unix de novos volumes (4 dígitos octal).	""

Exemplos de configurações

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

Configuração mínima

Esta é a configuração mínima absoluta de back-end. Com essa configuração, o Astra Trident descobre todos os pools de storage delegados ao Google Cloud NetApp volumes no local configurado e coloca novos volumes aleatoriamente em um desses pools. Como `nasType` é omitido, o `nfs` padrão se aplica e o back-end provisionará para volumes NFS.

Essa configuração é ideal quando você está apenas começando a usar o Google Cloud NetApp volumes e experimentando tudo. No entanto, na prática, é provável que você precise fornecer um escopo adicional para os volumes provisionados.

[illegible]

```
XsYg6gyxy4zq7OlwWgLwGa==  
-----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
```


[illegible]

```
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
```



```
znHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
znHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
znHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
XsYg6gyxy4zq7OlwWgLwGa==
-----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
```

O que se segue?

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

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

Para verificar se o back-end 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, algo está errado com a configuração do backend. Você pode descrever o back-end usando o `kubectl get tridentbackendconfig <backend-name>` comando ou visualizar os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs
```

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

Mais exemplos

Exemplos de definição 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` um "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 obter mais detalhes sobre classes de armazenamento, "[Crie uma classe de armazenamento](#)" consulte .

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	

Configure um back-end do Cloud Volumes Service para o Google Cloud

Saiba como configurar o NetApp Cloud Volumes Service para Google Cloud como back-end para sua instalação do Astra Trident usando as configurações de exemplo fornecidas.

Detalhes do driver do Google Cloud

O Astra Trident fornece ao `gcp-cvs` condutor a comunicação com o cluster. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Condutor	Protocolo	VolumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
gcp-cvs	NFS	Sistema de arquivos	RWO, ROX, RWX, RWOP	nfs

Saiba mais sobre o suporte ao Astra Trident para Cloud Volumes Service para Google Cloud

O Astra Trident pode criar volumes Cloud Volumes Service em um de dois "tipos de serviço":

- **CVS-Performance:** O tipo de serviço padrão Astra Trident. Esse tipo de serviço otimizado para performance é mais adequado para workloads de produção que valorizam a performance. O tipo de serviço CVS-Performance é uma opção de hardware que suporta volumes com um tamanho mínimo de 100 GiB. Você pode escolher um dos "três níveis de serviço":
 - `standard`
 - `premium`
 - `extreme`
- **CVS:** O tipo de serviço CVS fornece alta disponibilidade por zonas com níveis de desempenho limitados a moderados. O tipo de serviço CVS é uma opção de software que usa pools de armazenamento para dar suporte a volumes tão pequenos quanto 1 GiB. O pool de storage pode conter até 50 volumes em que todos os volumes compartilham a capacidade e a performance do pool. Você pode escolher um dos "dois níveis de serviço":
 - `standardsw`
 - `zoneredundantstandardsw`

O que você vai precisar

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

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

Opções de configuração de back-end

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

Parâmetro	Descrição	Padrão
<code>version</code>		Sempre 1
<code>storageDriverName</code>	Nome do controlador de armazenamento	"gcp-cvs"
<code>backendName</code>	Nome personalizado ou back-end de storage	Nome do driver e parte da chave da API
<code>storageClass</code>	Parâmetro opcional usado para especificar o tipo de serviço CVS. <code>software`</code> Use para selecionar o tipo de serviço CVS. Caso contrário, o Astra Trident assume o tipo de serviço CVS-Performance (<code>`hardware</code>).	
<code>storagePools</code>	Apenas tipo de serviço CVS. Parâmetro opcional usado para especificar pools de armazenamento para criação de volume.	
<code>projectNumber</code>	Número do projeto da conta Google Cloud. O valor é encontrado na página inicial do portal do Google Cloud.	
<code>hostProjectNumber</code>	Necessário se estiver usando uma rede VPC compartilhada. Neste cenário, <code>projectNumber</code> é o projeto de serviço, e <code>hostProjectNumber</code> é o projeto host.	
<code>apiRegion</code>	A região do Google Cloud onde o Astra Trident cria o Cloud Volumes Service volumes. Ao criar clusters de Kubernetes entre regiões, os volumes criados em um <code>apiRegion</code> podem ser usados em workloads programados em nós em várias regiões do Google Cloud. O tráfego entre regiões incorre em um custo adicional.	
<code>apiKey</code>	Chave de API para a conta de serviço do Google Cloud com a <code>netappcloudvolumes.admin</code> função. Ele inclui o conteúdo formatado em JSON do arquivo de chave privada de uma conta de serviço do Google Cloud (copiado literalmente no arquivo de configuração de back-end).	

Parâmetro	Descrição	Padrão
proxyURL	URL do proxy se o servidor proxy for necessário para se conectar à conta CVS. O servidor proxy pode ser um proxy HTTP ou um proxy HTTPS. Para um proxy HTTPS, a validação do certificado é ignorada para permitir o uso de certificados autoassinados no servidor proxy. Os servidores proxy com autenticação ativada não são suportados.	
nfsMountOptions	Controle refinado das opções de montagem NFS.	"3"
limitVolumeSize	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor.	"" (não aplicado por padrão)
serviceLevel	O nível de serviço CVS-Performance ou CVS para novos volumes. Os valores CVS-Performance são standard, premium, extreme ou . Os valores CVS são standardsw ou zoneredundantstandardsw.	O padrão CVS-Performance é "padrão". O padrão CVS é "standardsw".
network	Rede Google Cloud usada para Cloud Volumes Service volumes.	"predefinição"
debugTraceFlags	Debug flags para usar ao solucionar problemas. Exemplo, <code>\{"api":false, "method":true\}</code> . Não use isso a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	nulo
allowedTopologies	Para habilitar o acesso entre regiões, a definição do StorageClass para allowedTopologies deve incluir todas as regiões. Por exemplo: - key: topology.kubernetes.io/region values: - us-east1 - europe-west1	

Opções de provisionamento de volume

Você pode controlar o provisionamento de volume padrão defaults na seção 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 ou sub-redes IPv4 na notação CIDR.	"0,0.0,0/0"
snapshotDir	Acesso ao .snapshot diretório	"falso"
snapshotReserve	Porcentagem de volume reservado para snapshots	"" (aceitar o padrão CVS de 0)
size	O tamanho dos novos volumes. O mínimo de desempenho do CVS é de 100 GiB. CVS mínimo é de 1 GiB.	O tipo de serviço CVS-Performance é padrão para "100GiB". O tipo de serviço CVS não define um padrão, mas requer um mínimo de 1 GiB.

Exemplos de tipos de serviço CVS-Performance

Os exemplos a seguir fornecem exemplos de configurações para o tipo de serviço CVS-Performance.

Exemplo 1: Configuração mínima

Essa é a configuração mínima de back-end usando o tipo de serviço CVS-Performance padrão com o nível de serviço padrão.

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

Exemplo 2: Configuração do nível de serviço

Este exemplo ilustra as opções de configuração de back-end, incluindo nível de serviço e padrões de volume.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
proxyURL: http://proxy-server-hostname/
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 10Ti
serviceLevel: premium
defaults:
  snapshotDir: 'true'
  snapshotReserve: '5'
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  size: 5Ti
```

Exemplo 3: Configuração de pool virtual

Este exemplo usa `storage` para configurar pools virtuais e os `StorageClasses` que se referem a eles. [Definições de classe de armazenamento](#) Consulte para ver como as classes de armazenamento foram definidas.

Aqui, padrões específicos são definidos para todos os pools virtuais, que definem o `snapshotReserve` em 5% e o `exportRule` para 0,0.0,0/0. Os pools virtuais são definidos na `storage` seção. Cada pool virtual individual define seu próprio `serviceLevel`, e alguns pools substituem os valores padrão. Rótulos de pool virtual foram usados para diferenciar os pools com base em `performance` e `protection`.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
nfsMountOptions: vers=3,proto=tcp,timeo=600
defaults:
  snapshotReserve: '5'
  exportRule: 0.0.0.0/0
labels:
  cloud: gcp
region: us-west2
storage:
- labels:
  performance: extreme
  protection: extra
  serviceLevel: extreme
```

```

defaults:
  snapshotDir: 'true'
  snapshotReserve: '10'
  exportRule: 10.0.0.0/24
- labels:
  performance: extreme
  protection: standard
  serviceLevel: extreme
- labels:
  performance: premium
  protection: extra
  serviceLevel: premium
defaults:
  snapshotDir: 'true'
  snapshotReserve: '10'
- labels:
  performance: premium
  protection: standard
  serviceLevel: premium
- labels:
  performance: standard
  serviceLevel: standard

```

Definições de classe de armazenamento

As seguintes definições do StorageClass se aplicam ao exemplo de configuração de pool virtual. Usando `parameters.selector`o , você pode especificar para cada StorageClass o pool virtual usado para hospedar um volume. O volume terá os aspetos definidos no pool escolhido.

Exemplo de classe de armazenamento

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

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

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

Exemplos de tipo de serviço CVS

Os exemplos a seguir fornecem exemplos de configurações para o tipo de serviço CVS.

Exemplo 1: Configuração mínima

Essa é a configuração mínima de back-end usada `storageClass` para especificar o tipo de serviço CVS e o nível de serviço padrão `standardsw`.

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


Exemplo 2: Configuração do pool de armazenamento

Essa configuração de back-end de exemplo é usada `storagePools` para configurar um pool de armazenamento.

```
---
version: 1
storageDriverName: gcp-cvs
backendName: gcp-std-so-with-pool
projectNumber: '531265380079'
apiRegion: europe-west1
apiKey:
  type: service_account
  project_id: cloud-native-data
  private_key_id: "<id_value>"
  private_key: |-
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@cloud-native-
data.iam.gserviceaccount.com
  client_id: '107071413297115343396'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40cloud-native-data.iam.gserviceaccount.com
storageClass: software
zone: europe-west1-b
network: default
storagePools:
- 1bc7f380-3314-6005-45e9-c7dc8c2d7509
serviceLevel: Standardsw
```

O que se segue?

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

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

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

```
tridentctl logs
```

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

Configurar um back-end NetApp HCI ou SolidFire

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

Detalhes do driver do elemento

O Astra Trident fornece o `solidfire-san` driver de storage 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, o Astra Trident cria um volume e cria um sistema de arquivos. O tipo de sistema de arquivos é especificado pelo `StorageClass`.

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

Antes de começar

Você precisará do seguinte antes de criar um backend de elemento.

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

Opções de configuração de back-end

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

Parâmetro	Descrição	Padrão
version		Sempre 1
storageDriverName	Nome do controlador de armazenamento	Sempre "SolidFire-san"

Parâmetro	Descrição	Padrão
backendName	Nome personalizado ou back-end de storage	Endereço IP "SolidFire_" e armazenamento (iSCSI)
Endpoint	MVIP para o cluster SolidFire com credenciais de locatário	
SVIP	Porta e endereço IP de armazenamento (iSCSI)	
labels	Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes.	""
TenantName	Nome do locatário a utilizar (criado se não for encontrado)	
InitiatorIFace	Restringir o tráfego iSCSI a uma interface de host específica	"padrão"
UseCHAP	Use CHAP para autenticar iSCSI. O Astra Trident usa CHAP.	verdadeiro
AccessGroups	Lista de IDs de Grupo de Acesso a utilizar	Encontra a ID de um grupo de acesso chamado "Trident"
Types	Especificações de QoS	
limitVolumeSize	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor	"" (não aplicado por padrão)
debugTraceFlags	Debug flags para usar ao solucionar problemas. Por exemplo, "api":false, "método":true	nulo



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

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

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

```

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

```

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

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

O Astra Trident copia rótulos presentes em um pool de storage para a LUN de storage de back-end no provisionamento. Por conveniência, os administradores de storage podem definir rótulos por pool virtual e volumes de grupo por rótulo.

No arquivo de definição de back-end de exemplo mostrado abaixo, padrões específicos são definidos para todos os pools de armazenamento, que definem o `type` em Prata. Os pools virtuais são definidos na `storage` seção. Neste exemplo, alguns dos pools de armazenamento 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 do StorageClass referem-se aos pools virtuais acima. Usando o

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

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

```

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

```

Encontre mais informações

- ["Grupos de acesso de volume"](#)

Controladores SAN ONTAP

Descrição geral do controlador SAN ONTAP

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

Detalhes do driver SAN ONTAP

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



Se você estiver usando o Astra Control para proteção, recuperação e mobilidade, leia [Compatibilidade com driver Astra Control](#).

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

Condutor	Protocolo	VolumeMo de	Modos de acesso suportados	Sistemas de arquivos suportados
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 do sistema de arquivos.	xfs ext3, , ext4

Compatibilidade com driver Astra Control

O Astra Control oferece proteção aprimorada, recuperação de desastres e mobilidade (migrando volumes entre clusters Kubernetes) para volumes criados com os `ontap-nas` drivers, `ontap-nas-flexgroup` e `ontap-san`. ["Pré-requisitos de replicação do Astra Control"](#) Consulte para obter detalhes.



- Use `ontap-san-economy` somente se a contagem de uso de volume persistente for esperada ser maior que ["Limites de volume ONTAP suportados"](#).
- Use `ontap-nas-economy` somente se a contagem de uso de volume persistente for esperada para ser maior do que ["Limites de volume ONTAP suportados"](#) e o `ontap-san-economy` driver não puder ser usado.
- Não use o uso `ontap-nas-economy` se você antecipar a necessidade de proteção de dados, recuperação de desastres ou mobilidade.

Permissões do usuário

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



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

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

Considerações adicionais para NVMe/TCP

O Astra Trident dá suporte ao protocolo NVMe (non-volátil Memory Express) usando `ontap-san` o driver, incluindo:

- IPv6
- Snapshots e clones de volumes NVMe
- Redimensionamento de um volume NVMe
- Importação de um volume NVMe que foi criado fora do Astra Trident para que seu ciclo de vida possa ser gerenciado pelo Astra Trident
- Multipathing nativo NVMe
- Desligamento gracioso ou vergonhoso dos K8s nós (24,06)

O Astra Trident não é compatível com:

- DH-HMAC-CHAP que é suportado nativamente pelo NVMe
- Multipathing de mapeador de dispositivos (DM)
- Criptografia LUKS

Prepare-se para configurar o back-end com drivers SAN ONTAP

Entenda os requisitos e as opções de autenticação para configurar um back-end do ONTAP com drivers de SAN ONTAP.

Requisitos

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

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

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

Autenticar o back-end do ONTAP

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

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

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



Se você tentar fornecer **credenciais e certificados**, a criação de back-end falhará com um erro que mais de um método de autenticação foi fornecido no arquivo de configuração.

Ative a autenticação baseada em credenciais

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

Uma definição de backend de exemplo será assim:

YAML

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

JSON

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

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

Ativar autenticação baseada em certificado

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

- `ClientCertificate`: Valor codificado base64 do certificado do cliente.
- `ClientPrivateKey`: Valor codificado em base64 da chave privada associada.
- `TrustedCACertificate`: Valor codificado base64 do certificado CA confiável. Se estiver usando uma CA

confiável, esse parâmetro deve ser fornecido. Isso pode ser ignorado se nenhuma CA confiável for usada.

Um fluxo de trabalho típico envolve as etapas a seguir.

Passos

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

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

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

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

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

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

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

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

5. Teste a autenticação usando certificado gerado. Substitua o ONTAP Management LIF> e o <vserver name> por IP de LIF de gerenciamento e nome da SVM.

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

6. Codificar certificado, chave e certificado CA confiável com Base64.

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

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

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

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

Atualizar métodos de autenticação ou girar credenciais

Você pode atualizar um back-end existente para usar um método de autenticação diferente ou para girar suas credenciais. Isso funciona de ambas as maneiras: Backends que fazem uso de nome de usuário / senha podem ser atualizados para usar certificados; backends que utilizam certificados podem ser atualizados para nome de usuário / senha com base. Para fazer isso, 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
SanBackend	ontap-san	586b1cd5-8cf8-428d-a76c-2872713612c1

```

+-----+-----+-----+
+-----+-----+
| NAME | STORAGE DRIVER | UUID |
STATE | VOLUMES |
+-----+-----+-----+
+-----+-----+
| SanBackend | ontap-san | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online | 9 |
+-----+-----+-----+
+-----+-----+

```



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

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

Autentique conexões com CHAP bidirecional

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

```

---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz

```



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

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

Como funciona

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

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

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

Gire credenciais e atualize os backends

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



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

```
cat backend-san.json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "ontap_san_chap",
  "managementLIF": "192.168.0.135",
  "svm": "ontap_iscsi_svm",
  "useCHAP": true,
  "username": "vsadmin",
  "password": "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-4378-8a33-1a472ffbeeb5c |
online | 7 |
+-----+-----+-----+-----+
+-----+-----+
```

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

Exemplos e opções de configuração de SAN ONTAP

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

Opções de configuração de back-end

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

Parâmetro	Descrição	Padrão
version		Sempre 1
storageDrive rName	Nome do controlador de armazenamento	ontap-nas ontap-nas- economy, , ontap-nas- flexgroup ontap-san , , , ontap-san-economy
backendName	Nome personalizado ou back-end de storage	Nome do driver e dataLIF
managementLI F	Endereço IP de um cluster ou LIF de gerenciamento de SVM. Um nome de domínio totalmente qualificado (FQDN) pode ser especificado. Pode ser definido para usar endereços IPv6 se o Astra Trident tiver sido instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555] . Para o switchover MetroCluster otimizado, consulte o [mcc-best] .	"10,0,0,1", "[2001:1234:abcd::fefe]"
dataLIF	Endereço IP do protocolo LIF. Não especifique para iSCSI. O Astra Trident usa " Mapa de LUN seletivo da ONTAP " para descobrir os LIFs iSCSI necessários para estabelecer uma sessão de vários caminhos. Um aviso é gerado se dataLIF for definido explicitamente. Omita para MetroCluster. Consulte [mcc-best] .	Derivado do SVM
svm	Máquina virtual de armazenamento para usar omit for MetroCluster. Consulte [mcc-best] .	Derivado se uma SVM managementLIF for especificada
useCHAP	Use CHAP para autenticar iSCSI para drivers SAN ONTAP [Boolean]. Defina como true para o Astra Trident para configurar e usar CHAP bidirecional como a autenticação padrão para o SVM dado no back-end. " Prepare-se para configurar o back-end com drivers SAN ONTAP "Consulte para obter detalhes.	false
chapInitiato rSecret	Segredo do iniciador CHAP. Necessário se useCHAP=true	""
labels	Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes	""
chapTargetIn itiatorSecre t	Segredo do iniciador de destino CHAP. Necessário se useCHAP=true	""
chapUsername	Nome de utilizador de entrada. Necessário se useCHAP=true	""
chapTargetUs ername	Nome de utilizador alvo. Necessário se useCHAP=true	""
clientCertif icate	Valor codificado em base64 do certificado do cliente. Usado para autenticação baseada em certificado	""

Parâmetro	Descrição	Padrão
clientPrivateKey	Valor codificado em base64 da chave privada do cliente. Usado para autenticação baseada em certificado	""
trustedCACertificate	Valor codificado em base64 do certificado CA confiável. Opcional. Usado para autenticação baseada em certificado.	""
username	Nome de usuário necessário para se comunicar com o cluster ONTAP. Usado para autenticação baseada em credenciais.	""
password	Senha necessária para se comunicar com o cluster ONTAP. Usado para autenticação baseada em credenciais.	""
svm	Máquina virtual de armazenamento para usar	Derivado se uma SVM managementLIF for especificada
storagePrefix	Prefixo usado ao provisionar novos volumes na SVM. Não pode ser modificado mais tarde. Para atualizar esse parâmetro, você precisará criar um novo backend.	trident
limitAggregateUsage	Falha no provisionamento se o uso estiver acima dessa porcentagem. Se você estiver usando um back-end do Amazon FSX for NetApp ONTAP, não limitAggregateUsage`especifique . O fornecido `fsxadmin e vsadmin não contém as permissões necessárias para recuperar o uso agregado e limitá-lo usando o Astra Trident.	"" (não aplicado por padrão)
limitVolumeSize	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor. Também restringe o tamanho máximo dos volumes que gerencia para qtrees e LUNs.	"" (não aplicado por padrão)
lunsPerFlexvol	Máximo de LUNs por FlexVol, tem de estar no intervalo [50, 200]	100
debugTraceFlags	Debug flags para usar ao solucionar problemas. Por exemplo, não use a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	null

Parâmetro	Descrição	Padrão
useREST	Parâmetro booleano para usar APIs REST do ONTAP. useREST Quando definido como <code>true</code> , o Astra Trident usará as APIs REST do ONTAP para se comunicar com o back-end. Quando definido como <code>false</code> , o Astra Trident usará chamadas ZAPI do ONTAP para se comunicar com o back-end. Esse recurso requer o ONTAP 9.11,1 e posterior. Além disso, a função de login do ONTAP usada deve ter acesso ao <code>ontap</code> aplicativo. Isso é satisfeito com as funções <code>cluster-admin</code> predefinidas <code>vsadmin</code> . A partir da versão Astra Trident 24,06 e ONTAP 9.15,1 ou posterior, <code>useREST</code> é definida como <code>true</code> por padrão; altere <code>useREST</code> para <code>false</code> para usar chamadas ONTAP ZAPI. useREST É totalmente qualificado para NVMe/TCP.	<code>true</code> Para ONTAP 9.15,1 ou posterior, caso contrário <code>false</code> .
sanType	Utilize para selecionar <code>iscsi</code> para iSCSI ou <code>nvme</code> para NVMe/TCP.	<code>iscsi</code> se estiver em branco

Opções de configuração de back-end para volumes de provisionamento

Você pode controlar o provisionamento padrão usando essas opções na `defaults` seção da configuração. Para obter um exemplo, consulte os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
spaceAllocation	Alocação de espaço para LUNs	"verdadeiro"
spaceReserve	Modo de reserva de espaço; "nenhum" (fino) ou "volume" (grosso)	"nenhum"
snapshotPolicy	Política de instantâneos a utilizar	"nenhum"
qosPolicy	Grupo de políticas de QoS a atribuir aos volumes criados. Escolha uma das <code>qosPolicy</code> ou <code>adaptiveQosPolicy</code> por pool de armazenamento/backend. O uso de grupos de política de QoS com o Astra Trident requer o ONTAP 9.8 ou posterior. Recomendamos o uso de um grupo de políticas de QoS não compartilhado e garantir que o grupo de políticas seja aplicado individualmente a cada componente. Um grupo de política de QoS compartilhado aplicará o limite máximo da taxa de transferência total de todos os workloads.	""
adaptiveQosPolicy	Grupo de políticas de QoS adaptável a atribuir para volumes criados. Escolha uma das <code>qosPolicy</code> ou <code>adaptiveQosPolicy</code> por pool de armazenamento/backend	""

Parâmetro	Descrição	Padrão
snapshotReserve	Porcentagem de volume reservado para snapshots	"0" se snapshotPolicy for "nenhum", caso contrário ""
splitOnClone	Divida um clone de seu pai na criação	"falso"
encryption	Ative a criptografia de volume do NetApp (NVE) no novo volume; o padrão é false. O NVE deve ser licenciado e habilitado no cluster para usar essa opção. Se o NAE estiver ativado no back-end, qualquer volume provisionado no Astra Trident será o NAE ativado. Para obter mais informações, consulte: "Como o Astra Trident funciona com NVE e NAE" .	"falso"
luksEncryption	Ativar encriptação LUKS. "Usar a configuração de chave unificada do Linux (LUKS)" Consulte a . A criptografia LUKS não é compatível com NVMe/TCP.	""
securityStyle	Estilo de segurança para novos volumes	unix
tieringPolicy	Política de disposição em camadas para usar "nenhuma"	"Somente snapshot" para configuração pré-ONTAP 9.5 SVM-DR
nameTemplate	Modelo para criar nomes de volume personalizados.	""
limitVolumePoolSize	Tamanho máximo de FlexVol requestable ao usar LUNs no back-end ONTAP-san-econômico.	"" (não aplicado por padrão)

Exemplos de provisionamento de volume

Aqui está um exemplo com padrões 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'

```



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

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

$$\text{Total volume size} = [(\text{PVC requested size}) / (1 - (\text{snapshotReserve percentage} / 100))] * 1.1$$

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

Vserver	Volume	Aggregate	State	Type	Size	Available	Used%
		_pvc_89f1c156_3801_4de4_9f9d_034d54c395f4	online	RW	10GB	5.00GB	0%
		_pvc_e42ec6fe_3baa_4af6_996d_134adbbb8e6d	online	RW	5.79GB	5.50GB	0%
		_pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba	online	RW	1GB	511.8MB	0%

3 entries were displayed.

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

Exemplos mínimos de configuração

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



Se você estiver usando o Amazon FSX no NetApp ONTAP com Astra Trident, recomendamos que você especifique nomes DNS para LIFs em vez de endereços IP.

Exemplo de SAN ONTAP

Esta é uma configuração básica usando `ontap-san` o 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 de economia de SAN ONTAP

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

1. exemplo

Você pode configurar o back-end para evitar ter que atualizar manualmente a definição do back-end após o switchover e o switchback durante "[Replicação e recuperação da SVM](#)".

Para comutação e switchback contínuos, especifique o SVM usando `managementLIF` e omita os `dataLIF` parâmetros e. `svm` Por exemplo:

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

Exemplo de autenticação baseada em certificado

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

Esses exemplos criam um backend com useCHAP definido como true.

Exemplo de 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 CHAP de economia de SAN ONTAP

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


Exemplo de NVMe/TCP

Você precisa ter um SVM configurado com NVMe no back-end do ONTAP. Esta é uma configuração básica de back-end 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 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}}_{{.volume.RequestName}}"
},
"labels": {"cluster": "ClusterA", "PVC":
  "{{.volume.Namespace}}_{{.volume.RequestName}}"}
}
```

Exemplos de backends com pools virtuais

Nesses arquivos de definição de back-end de exemplo, padrões específicos são definidos para todos os pools de armazenamento, como `spaceReserve` em `nenhum`, `spaceAllocation` em `falso` e `encryption` em `falso`. Os pools virtuais são definidos na seção `armazenamento`.

O Astra Trident define rótulos de provisionamento no campo "Comentários". Os comentários são definidos no `FlexVol`. O Astra Trident copia todas as etiquetas presentes em um pool virtual para o volume de storage no provisionamento. Por conveniência, os administradores de storage podem definir rótulos por pool virtual e volumes de grupo por rótulo.

Nesses exemplos, alguns dos pools de armazenamento definem seus próprios `spaceReserve` `spaceAllocation` valores , e `encryption` , e alguns pools substituem os valores padrão.

Exemplo de SAN ONTAP



```

---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: 'false'
  encryption: 'false'
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
- labels:
  protection: gold
  creditpoints: '40000'
  zone: us_east_1a
  defaults:
    spaceAllocation: 'true'
    encryption: 'true'
    adaptiveQosPolicy: adaptive-extreme
- labels:
  protection: silver
  creditpoints: '20000'
  zone: us_east_1b
  defaults:
    spaceAllocation: 'false'
    encryption: 'true'
    qosPolicy: premium
- labels:
  protection: bronze
  creditpoints: '5000'
  zone: us_east_1c
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'

```

Exemplo de economia de SAN ONTAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: 'false'
  encryption: 'false'
labels:
  store: san_economy_store
region: us_east_1
storage:
- labels:
  app: oracledb
  cost: '30'
  zone: us_east_1a
  defaults:
    spaceAllocation: 'true'
    encryption: 'true'
- labels:
  app: postgresdb
  cost: '20'
  zone: us_east_1b
  defaults:
    spaceAllocation: 'false'
    encryption: 'true'
- labels:
  app: mysqldb
  cost: '10'
  zone: us_east_1c
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
- labels:
  department: legal
  creditpoints: '5000'
  zone: us_east_1c
```

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

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'
```

Mapeie os backends para StorageClasses

As seguintes definições do StorageClass referem-se ao [Exemplos de backends com pools virtuais](#). Usando o `parameters.selector` campo, cada StorageClass chama quais pools virtuais podem ser usados para hospedar um volume. O volume terá os aspetos definidos no pool virtual escolhido.

- O `protection-gold` StorageClass será mapeado para o primeiro pool virtual `ontap-san` no back-end. Esta é a única piscina que oferece proteção de nível dourado.

```
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 será mapeado para o segundo e terceiro pool virtual no `ontap-san` back-end. Estas são as únicas piscinas que oferecem um nível de proteção diferente do ouro.

```
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` back-end. Este é o único pool que oferece configuração de pool de armazenamento para o aplicativo 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` back-end. Esta é a única piscina que oferece proteção de nível de 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 será mapeado para o terceiro pool virtual no `ontap-san` back-end e o quarto pool virtual no `ontap-san-economy` back-end. Estas são as únicas ofertas de pool com 5000 pontos de crédito.

```

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`o . Esta é a única piscina 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"

```

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

Drivers nas ONTAP

Descrição geral do controlador ONTAP nas

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

Detalhes do driver nas do ONTAP

O Astra Trident fornece os seguintes drivers de storage nas para se comunicar com o cluster ONTAP. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMux* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).



Se você estiver usando o Astra Control para proteção, recuperação e mobilidade, leia [Compatibilidade com driver Astra Control](#).

Condutor	Protocolo	VolumeMo de	Modos de acesso suportados	Sistemas de arquivos suportados
ontap-nas	NFS, SMB	Sistema de arquivos	RWO, ROX, RWX, RWOP	"" nfs, , smb

Condutor	Protocolo	VolumeMo de	Modos de acesso suportados	Sistemas de arquivos suportados
ontap-nas-economy	NFS, SMB	Sistema de ficheiros	RWO, ROX, RWX, RWOP	"" nfs, , smb
ontap-nas-flexgroup	NFS, SMB	Sistema de ficheiros	RWO, ROX, RWX, RWOP	"" nfs, , smb

Compatibilidade com driver Astra Control

O Astra Control oferece proteção aprimorada, recuperação de desastres e mobilidade (migrando volumes entre clusters Kubernetes) para volumes criados com os `ontap-nas` drivers, `ontap-nas-flexgroup` e `ontap-san`. ["Pré-requisitos de replicação do Astra Control"](#) Consulte para obter detalhes.



- Use `ontap-san-economy` somente se a contagem de uso de volume persistente for esperada ser maior que ["Limites de volume ONTAP suportados"](#).
- Use `ontap-nas-economy` somente se a contagem de uso de volume persistente for esperada para ser maior do que ["Limites de volume ONTAP suportados"](#) e o `ontap-san-economy` driver não puder ser usado.
- Não use o uso `ontap-nas-economy` se você antecipar a necessidade de proteção de dados, recuperação de desastres ou mobilidade.

Permissões do usuário

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

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



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

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

Prepare-se para configurar um back-end com drivers nas ONTAP

Entenda os requisitos, as opções de autenticação e as políticas de exportação para configurar um back-end do ONTAP com drivers nas do ONTAP.

Requisitos

- Para todos os back-ends ONTAP, o Astra Trident requer pelo menos um agregado atribuído ao SVM.
- Você pode executar mais de um driver e criar classes de armazenamento que apontam para um ou outro. Por exemplo, você pode configurar uma classe Gold que usa o `ontap-nas` driver e uma classe Bronze que usa o `ontap-nas-economy` um.
- Todos os seus nós de trabalho do Kubernetes precisam ter as ferramentas NFS apropriadas instaladas. ["aqui"](#)Consulte para obter mais detalhes.
- O Astra Trident é compatível com volumes SMB montados em pods executados apenas em nós do Windows. [Prepare-se para provisionar volumes SMB](#)Consulte para obter detalhes.

Autenticar o back-end do ONTAP

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

- Baseado em credenciais: Esse modo requer permissões suficientes para o back-end 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: Esse modo requer que um certificado seja instalado no back-end para que o Astra Trident se comunique com um cluster ONTAP. Aqui, a definição de back-end deve conter valores codificados em Base64 do certificado de cliente, chave e certificado de CA confiável, se usado (recomendado).

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



Se você tentar fornecer **credenciais e certificados**, a criação de back-end falhará com um erro que mais de um método de autenticação foi fornecido no arquivo de configuração.

Ative a autenticação baseada em credenciais

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

Uma definição de backend de exemplo será assim:

YAML

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

JSON

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

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

Ativar autenticação baseada em certificado

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

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

Um fluxo de trabalho típico envolve as etapas a seguir.

Passos

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

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

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

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

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

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

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

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

5. Teste a autenticação usando certificado gerado. Substitua o ONTAP Management LIF> e o <vserver name> por IP de LIF de gerenciamento e nome da SVM. Você deve garantir que o LIF tenha sua política de serviço definida como default-data-management.

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

6. Codificar certificado, chave e certificado CA confiável com Base64.

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

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

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

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

Atualizar métodos de autenticação ou girar credenciais

Você pode atualizar um back-end existente para usar um método de autenticação diferente ou para girar suas credenciais. Isso funciona de ambas as maneiras: Backends que fazem uso de nome de usuário / senha podem ser atualizados para usar certificados; backends que utilizam certificados podem ser atualizados para nome de usuário / senha com base. Para fazer isso, 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 |
+-----+-----+-----+
+-----+-----+
```



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

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

Gerenciar políticas de exportação de NFS

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

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

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

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

Gerencie dinamicamente políticas de exportação

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



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

Exemplo

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

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



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

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

- `autoExportPolicy` está definido como `true`. Isso indica que o Astra Trident criará uma política de exportação para `svm1` o SVM e tratará da adição e exclusão de regras usando `autoExportCIDRs` blocos de endereço. Por exemplo, um back-end com UUID 403b5326-8482-40dB-96d0-d83fb3f4daec e `autoExportPolicy` definido como `true` cria uma política de exportação nomeada `trident-403b5326-8482-40db-96d0-d83fb3f4daec` no SVM.
- `autoExportCIDRs` contém uma lista de blocos de endereços. Este campo é opcional e o padrão é `["0.0.0.0/0", "::/0"]`. Se não estiver definido, o Astra Trident adiciona todos os endereços unicast de escopo

global encontrados nos nós de trabalho.

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

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

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

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

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

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

Para backends existentes anteriormente, a atualização do back-end com `tridentctl update backend` garantirá que o Astra Trident gerencie as políticas de exportação automaticamente. Isso criará uma nova

política de exportação nomeada após o UUID do backend e os volumes que estão presentes no backend usarão a política de exportação recém-criada quando forem montados novamente.



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

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

Prepare-se para provisionar volumes SMB

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



Você precisa configurar os protocolos NFS e SMB/CIFS na SVM para criar um `ontap-nas-economy` volume SMB para ONTAP no local. A falha na configuração desses protocolos fará com que a criação de volume SMB falhe.

Antes de começar

Antes de provisionar volumes SMB, você deve ter o seguinte:

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

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

- Um proxy CSI configurado como um serviço Windows. Para configurar um `csi-proxy`, "[GitHub: CSI Proxy](#)" consulte ou "[GitHub: CSI Proxy para Windows](#)" para nós do Kubernetes executados no Windows.

Passos

1. Para o ONTAP no local, você pode criar, opcionalmente, um compartilhamento SMB ou o Astra Trident pode criar um para você.



Compartilhamentos SMB são necessários para o Amazon FSX for ONTAP.

Você pode criar os compartilhamentos de administração SMB de duas maneiras usando o "[Microsoft Management Console](#)" snap-in pastas compartilhadas ou usando a CLI do ONTAP. Para criar compartilhamentos SMB usando a CLI do ONTAP:

- a. Se necessário, crie a estrutura do caminho do diretório para o compartilhamento.

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

- b. Crie um compartilhamento SMB associado ao SVM especificado:

```
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
```



"[Crie um compartilhamento SMB](#)" Consulte para obter detalhes completos.

2. Ao criar o back-end, você deve configurar o seguinte para especificar volumes SMB. Para obter todas as opções de configuração de back-end do FSX for ONTAP, "[Opções e exemplos de configuração do FSX for ONTAP](#)" consulte .

Parâmetro	Descrição	Exemplo
smbShare	Você pode especificar uma das seguintes opções: O nome de um compartilhamento SMB criado usando o Console de Gerenciamento da Microsoft ou a CLI do ONTAP; um nome para permitir que o Astra Trident crie o compartilhamento SMB; ou você pode deixar o parâmetro em branco para impedir o acesso comum ao compartilhamento aos volumes. Esse parâmetro é opcional para o ONTAP no local. Esse parâmetro é necessário para backends do Amazon FSX for ONTAP e não pode ficar em branco.	smb-share
nasType	Tem de estar definido para smb. Se 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 para volumes SMB
unixPermissions	Modo para novos volumes. Deve ser deixado vazio para volumes SMB.	""

Exemplos e opções de configuração do ONTAP nas

Aprenda a criar e usar drivers nas ONTAP com sua instalação do Astra Trident. Esta seção fornece exemplos de configuração de back-end e detalhes para mapear backends para StorageClasses.

Opções de configuração de back-end

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

Parâmetro	Descrição	Padrão
version		Sempre 1
storageDriverName	Nome do controlador de armazenamento	"ONTAP-nas", "ONTAP-nas-economy", "ONTAP-nas-FlexGroup", "ONTAP-san", "ONTAP-san-economy"
backendName	Nome personalizado ou back-end de storage	Nome do driver e 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 definido para usar endereços IPv6 se o Astra Trident tiver sido instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555] . Para o switchover MetroCluster otimizado, consulte o [mcc-best] .	"10,0,0,1", "[2001:1234:abcd::fefe]"
dataLIF	Endereço IP do protocolo LIF. Recomendamos especificar dataLIF. Se não for fornecido, o Astra Trident obtém LIFs de dados do SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) a ser usado para as operações de montagem NFS, permitindo que você crie um DNS de round-robin para balanceamento de carga em vários LIFs de dados. Pode ser alterado após a definição inicial. Consulte a . Pode ser definido para usar endereços IPv6 se o Astra Trident tiver sido instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555] . Omita para MetroCluster. Consulte [mcc-best] .	Endereço especificado ou derivado do SVM, se não for especificado (não recomendado)
svm	Máquina virtual de armazenamento para usar omit for MetroCluster . Consulte [mcc-best] .	Derivado se uma SVM managementLIF for especificada
autoExportPolicy	Ativar a criação e atualização automática da política de exportação [Boolean]. Com autoExportPolicy as opções e autoExportCIDsRs, o Astra Trident pode gerenciar políticas de exportação automaticamente.	falso
autoExportCIDsRs	Lista de CIDR para filtrar IPs de nós do Kubernetes quando autoExportPolicy está ativado. Com autoExportPolicy as opções e autoExportCIDsRs, o Astra Trident pode gerenciar políticas de exportação automaticamente.	["0,0,0,0/0", ":::0"]»
labels	Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes	""
clientCertificate	Valor codificado em base64 do certificado do cliente. Usado para autenticação baseada em certificado	""

Parâmetro	Descrição	Padrão
clientPrivateKey	Valor codificado em base64 da chave privada do cliente. Usado para autenticação baseada em certificado	""
trustedCACertificate	Valor codificado em base64 do certificado CA confiável. Opcional. Usado para autenticação baseada em certificado	""
username	Nome de usuário para se conectar ao cluster/SVM. Usado para autenticação baseada em credenciais	
password	Senha para se conectar ao cluster/SVM. Usado para autenticação baseada em credenciais	
storagePrefix	Prefixo usado ao provisionar novos volumes na SVM. Não pode ser atualizado depois de configurá-lo	"Trident"
limitAggregateUsage	Falha no provisionamento se o uso estiver acima dessa porcentagem. Não se aplica ao Amazon FSX for ONTAP	"" (não aplicado por padrão)
limitVolumeSize	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor. 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.	"" (não aplicado por padrão)
lunsPerFlexvol	Máximo de LUNs por FlexVol, tem de estar no intervalo [50, 200]	"100"
debugTraceFlags	Debug flags para usar ao solucionar problemas. Por exemplo, não use debugTraceFlags a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	nulo
nasType	Configurar a criação de volumes NFS ou SMB. As opções são nfs, smb ou null. A configuração como null padrão para volumes NFS.	nfs
nfsMountOptions	Lista separada por vírgulas de opções de montagem NFS. As opções de montagem para volumes persistentes do Kubernetes normalmente são especificadas em classes de storage, mas se nenhuma opção de montagem for especificada em uma classe de storage, o Astra Trident voltará a usar as opções de montagem especificadas no arquivo de configuração do back-end de storage. Se nenhuma opção de montagem for especificada na classe de storage ou no arquivo de configuração, o Astra Trident não definirá nenhuma opção de montagem em um volume persistente associado.	""
qtreesPerFlexvol	Qtrees máximos por FlexVol, têm de estar no intervalo [50, 300]	"200"

Parâmetro	Descrição	Padrão
smbShare	Você pode especificar uma das seguintes opções: O nome de um compartilhamento SMB criado usando o Console de Gerenciamento da Microsoft ou a CLI do ONTAP; um nome para permitir que o Astra Trident crie o compartilhamento SMB; ou você pode deixar o parâmetro em branco para impedir o acesso comum ao compartilhamento aos volumes. Esse parâmetro é opcional para o ONTAP no local. Esse parâmetro é necessário para backends do Amazon FSX for ONTAP e não pode ficar em branco.	smb-share
useREST	Parâmetro booleano para usar APIs REST do ONTAP. <code>useREST</code> Quando definido como <code>true</code> , o Astra Trident usará as APIs REST do ONTAP para se comunicar com o back-end. Quando definido como <code>false</code> , o Astra Trident usará chamadas ZAPI do ONTAP para se comunicar com o back-end. Esse recurso requer o ONTAP 9.11,1 e posterior. Além disso, a função de login do ONTAP usada deve ter acesso ao <code>ontap</code> aplicativo. Isso é satisfeito com as funções <code>e cluster-admin</code> predefinidas <code>vsadmin</code> . A partir da versão Astra Trident 24,06 e ONTAP 9.15,1 ou posterior, <code>useREST</code> é definida como <code>true</code> por padrão; altere <code>useREST</code> para <code>false</code> para usar chamadas ONTAP ZAPI.	<code>true</code> Para ONTAP 9.15,1 ou posterior, caso contrário <code>false</code> .
limitVolumePoolSize	Tamanho máximo de FlexVol requestable ao usar <code>qtrees</code> no backend ONTAP-nas-Economy.	"" (não aplicado por padrão)

Opções de configuração de back-end para volumes de provisionamento

Você pode controlar o provisionamento padrão usando essas opções na `defaults` seção da configuração. Para obter um exemplo, consulte os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
spaceAllocation	Alocação de espaço para LUNs	"verdadeiro"
spaceReserve	Modo de reserva de espaço; "nenhum" (fino) ou "volume" (grosso)	"nenhum"
snapshotPolicy	Política de instantâneos a utilizar	"nenhum"
qosPolicy	Grupo de políticas de QoS a atribuir aos volumes criados. Escolha uma das <code>qosPolicy</code> ou <code>adaptiveQosPolicy</code> por pool de armazenamento/backend	""

Parâmetro	Descrição	Padrão
<code>adaptiveQosPolicy</code>	Grupo de políticas de QoS adaptável a atribuir para volumes criados. Escolha uma das <code>qosPolicy</code> ou <code>adaptiveQosPolicy</code> por pool de armazenamento/backend. Não suportado pela ONTAP-nas-Economy.	""
<code>snapshotReserve</code>	Porcentagem de volume reservado para snapshots	"0" se <code>snapshotPolicy</code> for "nenhum", caso contrário ""
<code>splitOnClone</code>	Divida um clone de seu pai na criação	"falso"
<code>encryption</code>	Ative a criptografia de volume do NetApp (NVE) no novo volume; o padrão é <code>false</code> . O NVE deve ser licenciado e habilitado no cluster para usar essa opção. Se o NAE estiver ativado no back-end, qualquer volume provisionado no Astra Trident será o NAE ativado. Para obter mais informações, consulte: "Como o Astra Trident funciona com NVE e NAE" .	"falso"
<code>tieringPolicy</code>	Política de disposição em camadas para usar "nenhuma"	"Somente snapshot" para configuração pré-ONTAP 9.5 SVM-DR
<code>unixPermissions</code>	Modo para novos volumes	"777" para volumes NFS; vazio (não aplicável) para volumes SMB
<code>snapshotDir</code>	Controla o acesso ao <code>.snapshot</code> diretório	"falso"
<code>exportPolicy</code>	Política de exportação a utilizar	"predefinição"
<code>securityStyle</code>	Estilo de segurança para novos volumes. Estilos de segurança e <code>unix</code> suporte de NFS <code>mixed</code> . Suporta SMB <code>mixed</code> e <code>ntfs</code> estilos de segurança.	O padrão NFS é <code>unix</code> . O padrão SMB é <code>ntfs</code> .
<code>nameTemplate</code>	Modelo para criar nomes de volume personalizados.	""



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

Exemplos de provisionamento de volume

Aqui está um exemplo com padrões definidos:

```

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

```

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

O cálculo é o seguinte:

```

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

```

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

Vserver	Volume	Aggregate	State	Type	Size	Available	Used%
		_pvc_89f1c156_3801_4de4_9f9d_034d54c395f4	online	RW	10GB	5.00GB	0%
		_pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba	online	RW	1GB	511.8MB	0%

2 entries were displayed.

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

Exemplos mínimos de configuração

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



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

Exemplo de economia nas do ONTAP

```
---
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 ONTAP nas FlexGroup

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


Exemplo de MetroCluster

Você pode configurar o back-end para evitar ter que atualizar manualmente a definição do back-end após o switchover e o switchback durante ["Replicação e recuperação da SVM"](#)o .

Para comutação e switchback contínuos, especifique o SVM usando `managementLIF` e omita os `dataLIF` parâmetros 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 de configuração de back-end mínimo. `clientCertificate`, `clientPrivateKey` e `trustedCACertificate` (opcional, se estiver usando CA confiável) são preenchidos em `backend.json` e recebem os valores codificados em base64 do certificado do cliente, da chave privada e do certificado de 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 o Astra Trident a usar políticas de exportação dinâmicas para criar e gerenciar a política de exportação automaticamente. Isso funciona da mesma forma para os `ontap-nas-economy drivers` e `ontap-nas-flexgroup`.

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

Exemplo do Amazon FSX para ONTAP usando volumes SMB

O smbShare parâmetro é necessário para o FSX for 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}}_{{.volume.R
    equestName}}"
},
"labels": {"cluster": "ClusterA", "PVC":
  "{{.volume.Namespace}}_{{.volume.RequestName}}"}
}
```

Exemplos de backends com pools virtuais

Nos arquivos de definição de back-end de exemplo mostrados abaixo, padrões específicos são definidos para todos os pools de armazenamento, como `spaceReserve` em `nenhum`, `spaceAllocation` em `falso` e `encryption` em `falso`. Os pools virtuais são definidos na seção armazenamento.

O Astra Trident define rótulos de provisionamento no campo "Comentários". Os comentários são definidos no FlexVol for `ontap-nas` ou no FlexGroup `ontap-nas-flexgroup` for . O Astra Trident copia todas as etiquetas presentes em um pool virtual para o volume de storage no provisionamento. Por conveniência, os administradores de storage podem definir rótulos por pool virtual e volumes de grupo por rótulo.

Nesses exemplos, alguns dos pools de armazenamento definem seus próprios `spaceReserve` `spaceAllocation` valores , e `encryption` , 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 nas do ONTAP

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

```
encryption: 'false'
unixPermissions: '0775'
```

Mapeie os backends para StorageClasses

As seguintes definições do StorageClass referem-se [Exemplos de backends com pools virtuais](#) . Usando o `parameters.selector` campo, cada StorageClass chama quais pools virtuais podem ser usados para hospedar um volume. O volume terá os aspetos definidos no pool virtual escolhido.

- O `protection-gold` StorageClass será mapeado para o primeiro e segundo pool virtual `ontap-nas-flexgroup` no back-end. Estas são as únicas piscinas que oferecem proteção de nível de 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 será mapeado para o terceiro e quarto pool virtual no `ontap-nas-flexgroup` back-end. Estas são as únicas piscinas que oferecem um nível de proteção diferente do ouro.

```
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 `ontap-nas` no back-end. Este é o único pool que oferece configuração de pool de armazenamento para o aplicativo 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` back-end. Esta é a única piscina que oferece proteção de nível de 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 será mapeado para o terceiro pool virtual `ontap-nas` no back-end e o segundo pool virtual `ontap-nas-economy` no back-end. Estas são as únicas ofertas de pool com 5000 pontos de crédito.

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

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

Atualização `dataLIF` após a configuração inicial

Você pode alterar o LIF de dados após a configuração inicial executando o seguinte comando para fornecer o novo arquivo JSON de back-end com LIF de dados atualizado.

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



Se os PVCs estiverem anexados a um ou vários pods, você deverá reduzir todos os pods correspondentes e restaurá-los para que o novo LIF de dados entre em vigor.

Amazon FSX para NetApp ONTAP

Use o Astra Trident com o Amazon FSX para NetApp ONTAP

"[Amazon FSX para NetApp ONTAP](#)" É um serviço AWS totalmente gerenciado que permite que os clientes iniciem e executem sistemas de arquivos equipados com o sistema operacional de storage NetApp ONTAP. O FSX para ONTAP permite que você aproveite os recursos, o desempenho e os recursos administrativos do NetApp com os quais você já conhece, ao mesmo tempo em que aproveita a simplicidade, a agilidade, a segurança e a escalabilidade do armazenamento de dados na AWS. O FSX para ONTAP oferece suporte aos recursos do sistema de arquivos ONTAP e APIs de administração.

Você pode integrar seu sistema de arquivos do Amazon FSX for NetApp ONTAP ao Astra Trident para garantir que os clusters do Kubernetes executados no Amazon Elastic Kubernetes Service (EKS) possam provisionar volumes persistentes de bloco e arquivo com o respaldo do ONTAP.

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

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

Requisitos

Além "[Requisitos do Astra Trident](#)" do , para integrar o FSX para ONTAP com Astra Trident, você precisa de:

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



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

Considerações

- Volumes SMB:

- Os volumes SMB são suportados usando `ontap-nas` apenas o driver.
- Os volumes SMB não são compatíveis com o complemento Astra Trident EKS.
- O Astra Trident é compatível com volumes SMB montados em pods executados apenas em nós do Windows. "[Prepare-se para provisionar volumes SMB](#)" Consulte para obter detalhes.
- Antes do Astra Trident 24,02, os volumes criados nos sistemas de arquivos do Amazon FSX que têm backups automáticos ativados, não puderam ser excluídos pelo Trident. Para evitar esse problema no Astra Trident 24,02 ou posterior, especifique o `fsxFileSystemID`, `apiRegion` AWS , `AWS apikey` e `AWS secretKey` no arquivo de configuração de back-end do AWS FSX for ONTAP.



Se você estiver especificando uma função do IAM para o Astra Trident, poderá omitir especificar explicitamente os `apiRegion` campos , `apiKey` e `secretKey` para o Astra Trident. Para obter mais informações, "[Opções e exemplos de configuração do FSX for ONTAP](#)" consulte .

Autenticação

O Astra Trident oferece dois modos de autenticação.

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



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

- Baseado em certificado: O Astra Trident se comunicará com o SVM em seu sistema de arquivos FSX usando um certificado instalado no seu SVM.

Para obter detalhes sobre como ativar a autenticação, consulte a autenticação do tipo de driver:

- "[Autenticação nas ONTAP](#)"
- "[Autenticação SAN ONTAP](#)"

Encontre mais informações

- "[Documentação do Amazon FSX para NetApp ONTAP](#)"
- "[Blog post no Amazon FSX for NetApp ONTAP](#)"

Crie uma função do IAM e o AWS Secret

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



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

Crie o segredo do AWS Secret Manager

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

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

Criar política do IAM

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 Secret manager"
```

Policy JSON file:

```
policy.json:
{
  "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 e função IAM para a conta de serviço

O exemplo a seguir cria uma função do IAM para a 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 arn:aws:iam::aws:policy/service-
role/AmazonFSxNCSIDriverPolicy --approve
```

Instale o Astra Trident

O Astra Trident simplifica o gerenciamento de armazenamento do Amazon FSX for NetApp ONTAP no Kubernetes para permitir que seus desenvolvedores e administradores se concentrem na implantação de aplicações.

Você pode instalar o Astra Trident usando um dos seguintes métodos:

- Leme
- Complemento EKS

If you want to make use of the snapshot functionality, install the CSI snapshot controller add-on. Refer to <https://docs.aws.amazon.com/eks/latest/userguide/csi-snapshot-controller.html>.

Instale o Astra Trident através do leme

1. Faça o download do pacote de instalação do Astra Trident

O pacote de instalação do Astra Trident contém tudo o que você precisa para implantar o operador Trident e instalar o Astra Trident. Baixe e extraia a versão mais recente do instalador Astra Trident da seção Assets no GitHub.

```
wget https://github.com/NetApp/trident/releases/download/v24.06.0/trident-
installer-24.06.0.tar.gz
tar -xf trident-installer-24.06.0.tar.gz
cd trident-installer
```

2. Defina os valores para os sinalizadores **provedor de nuvem** e **identidade de nuvem** usando as seguintes variáveis de ambiente:

```
export CP="AWS"
export CI="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'"
```

O exemplo a seguir instala o Astra Trident e define o cloud-provider sinalizador como \$CP, e cloud-identity como \$CI:

```
helm install trident trident-operator-100.2406.0.tgz --set
cloudProvider=$CP --set cloudIdentity=$CI --namespace trident
```

Você pode usar o `helm list` comando para revisar detalhes de instalação, como nome, namespace, gráfico, status, versão do aplicativo e número de 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-100.2406.1	24.06.1

Instale o Astra Trident através do complemento EKS

O complemento Astra Trident EKS inclui os patches de segurança mais recentes, correções de bugs e é validado pela AWS para funcionar com o Amazon EKS. O complemento EKS permite que você garanta consistentemente que seus clusters do Amazon EKS estejam seguros e estáveis e reduza a quantidade de trabalho que você precisa fazer para instalar, configurar e atualizar complementos.

Pré-requisitos

Antes de configurar o complemento Astra Trident para AWS EKS, verifique se você tem o seguinte:

- Uma conta de cluster do Amazon EKS com assinatura complementar
- Permissões da AWS para o marketplace da AWS:
"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 existente do Amazon FSX for NetApp ONTAP

Habilite o complemento Astra Trident para AWS

Cluster DE EKS

Os seguintes comandos de exemplo instalam o complemento Astra Trident EKS:

```
eksctl create addon --cluster clusterName --name netapp_trident-operator  
--version v24.6.1-eksbuild  
eksctl create addon --cluster clusterName --name netapp_trident-operator  
--version v24.6.1-eksbuild.1 (com uma versão dedicada)
```



Ao configurar o parâmetro opcional `cloudIdentity`, certifique-se de especificar `cloudProvider` durante a instalação do Trident usando o complemento EKS.

Console de gerenciamento

1. Abra o console do Amazon EKS em <https://console.aws.amazon.com/eks/home#/clusters>.
2. No painel de navegação esquerdo, clique em **clusters**.
3. Clique no nome do cluster para o qual você deseja configurar o complemento NetApp Trident CSI.
4. Clique em **Add-ons** e, em seguida, clique em **Get more add-ons**.
5. Na página **S*elecionar add-ons**, faça o seguinte:
 - a. Na seção addons do AWS Marketplace, marque a caixa de seleção **Astra Trident by NetApp**.
 - b. Clique em **seguinte**.
6. Na página de configurações **Configure Selected add-ons**, faça o seguinte:
 - a. Selecione a **versão** que você gostaria de usar.
 - b. Para **Selecione função IAM**, deixe em **não definido**.
 - c. Expanda as **Configurações opcionais de configuração**, siga o esquema de configuração **Add-on** e defina o parâmetro `configurationValues` na seção **valores de configuração** para a função-arn que você criou na etapa anterior (o valor deve estar no seguinte formato:
`eks.amazonaws.com/role-arn:arn:aws:iam::464262061435:role/AmazonEKS_FSXN_CSI_DriverRole`). Se você selecionar **Substituir** para o método de resolução de conflitos, uma ou mais configurações para o suplemento existente podem ser sobrescritas com as configurações de complemento do Amazon EKS. Se você não ativar essa 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 essa opção, certifique-se de que o complemento do Amazon EKS não gerencie as configurações que você precisa para gerenciar automaticamente.



Ao configurar o parâmetro opcional `cloudIdentity`, certifique-se de especificar `cloudProvider` durante a instalação do Trident usando o complemento EKS.

7. Escolha **seguinte**.
8. Na página **Revisão e adição**, escolha **criar**.

Depois que a instalação do complemento estiver concluída, você verá o complemento instalado.

CLI DA AWS

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

```
add-on.json
{
    "clusterName": "<eks-cluster>",
    "addonName": "netapp_trident-operator",
    "addonVersion": "v24.6.1-eksbuild.1",
    "serviceAccountRoleArn": "arn:aws:iam::123456:role/astratrident-
role",
    "configurationValues": "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/astratrident-
role'\",
    \"cloudProvider\": \"AWS\"}"
}
```



Ao configurar o parâmetro opcional `cloudIdentity`, certifique-se de especificar `AWS` como o `cloudProvider` durante a instalação do Trident usando o complemento EKS.

2. Instalar o complemento Astra Trident EKS

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

Atualize o complemento Astra Trident EKS

Cluster DE EKS

- Verifique a versão atual do seu complemento FSxN Trident CSI. Substitua `my-cluster` pelo nome do 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	v24.6.1-eksbuild.1	ACTIVE	0
{ "cloudIdentity": "'eks.amazonaws.com/role-arn:arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'" }			

- Atualize o complemento para a versão retornada em ATUALIZAÇÃO DISPONÍVEL na saída da etapa anterior.

```
eksctl update addon --name netapp_trident-operator --version v24.6.1-eksbuild.1 --cluster my-cluster --force
```

Se você remover `--force` a opção e qualquer uma das configurações de complemento do Amazon EKS entrar em conflito com as configurações existentes, a atualização do complemento do Amazon EKS falhará; você receberá uma mensagem de erro para ajudá-lo a resolver o conflito. Antes de especificar essa opção, verifique se o complemento do Amazon EKS não gerencia as configurações que você precisa gerenciar, pois essas configurações são sobrescritas com essa opção. Para obter mais informações sobre outras opções para essa configuração, "[Complementos](#)" consulte . Para obter mais informações sobre o gerenciamento de campo do Amazon EKS Kubernetes, "[Gerenciamento de campo do Kubernetes](#)" consulte .

Console de gerenciamento

1. Abra o console do Amazon EKS <https://console.aws.amazon.com/eks/home#/clusters> .
2. No painel de navegação esquerdo, clique em **clusters**.
3. Clique no nome do cluster para o qual você deseja atualizar o complemento NetApp Trident CSI.
4. Clique na guia **Complementos**.
5. Clique em **Astra Trident by NetApp** e, em seguida, clique em **Editar**.
6. Na página **Configurar o Astra Trident by NetApp**, faça o seguinte:
 - a. Selecione a **versão** que você gostaria de usar.
 - b. (Opcional) você pode expandir as **Configurações opcionais de configuração** e modificar conforme necessário.
 - c. Clique em **Salvar alterações**.

CLI DA AWS

O exemplo a seguir atualiza o complemento EKS:

```
aws eks update-addon --cluster-name my-cluster netapp_trident-operator vpc-cni  
--addon-version v24.6.1-eksbuild.1 \  
--service-account-role-arn arn:aws:iam::111122223333:role/role-name
```

```
--configuration-values '{}' --resolve-conflicts --preserve
```

Desinstale/remova o complemento Astra Trident EKS

Você tem duas opções para remover um complemento do Amazon EKS:

- **Preserve o software complementar no cluster** – essa opção remove o gerenciamento do Amazon EKS de qualquer configuração. Ele também remove a capacidade do Amazon EKS de notificá-lo de atualizações e atualizar automaticamente o complemento do Amazon EKS depois de iniciar uma atualização. No entanto, ele preserva o software complementar no cluster. Essa opção torna o complemento uma instalação autogerenciada, em vez de um complemento do Amazon EKS. Com essa opção, não há tempo de inatividade para o complemento. Guarde a `--preserve` opção no comando para preservar o complemento.
- **Remover software complementar inteiramente do cluster** – recomendamos que você remova o suplemento do Amazon EKS do cluster somente se não houver recursos no cluster que dependam dele. Remova `--preserve` a opção do `delete` comando para remover o complemento.



Se o complemento tiver uma conta do IAM associada a ele, a conta do IAM não será removida.

Cluster DE EKS

O comando a seguir desinstala o complemento Astra Trident EKS:

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

Console de gerenciamento

1. Abra o console do Amazon EKS em <https://console.aws.amazon.com/eks/home#/clusters>.
2. No painel de navegação esquerdo, clique em **clusters**.
3. Clique no nome do cluster para o qual você deseja remover o complemento NetApp Trident CSI.
4. Clique na guia **Complementos** e, em seguida, clique em **Astra Trident by NetApp**.*
5. Clique em **Remover**.
6. Na caixa de diálogo **Remover NetApp_Trident-operator confirmation**, faça o seguinte:
 - a. Se você quiser que o Amazon EKS pare de gerenciar as configurações do complemento, selecione **Preserve on cluster**. Faça isso se quiser manter o software complementar no cluster para que você possa gerenciar todas as configurações do complemento por conta própria.
 - b. Digite **NetApp_Trident-operator**.
 - c. Clique em **Remover**.

CLI DA AWS

Substitua `my-cluster` pelo nome do cluster e execute o seguinte comando.

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

Configure o back-end de armazenamento

Integração de driver SAN e nas ONTAP

Você pode criar um arquivo de back-end usando as credenciais SVM (nome de usuário e senha) armazenadas no AWS Secret Manager, conforme mostrado neste exemplo:

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
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:xxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
  },
  "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:xxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

Para obter informações sobre como criar backends, consulte estas páginas:

- ["Configurar um back-end com drivers nas ONTAP"](#)
- ["Configure um back-end com drivers SAN ONTAP"](#)

Detalhes do driver FSX for ONTAP

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

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

Para obter informações sobre o condutor, ["Controladores NAS"](#) consulte e ["Controladores SAN"](#).

Exemplos de configurações

Configuração do AWS FSX for ONTAP com gerenciador de segredos

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

Configuração da classe de armazenamento 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 ativo Directory. Os volumes SMB são suportados usando `ontap-nas` apenas o driver.

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

Configuração avançada de backend e exemplos

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

Parâmetro	Descrição	Exemplo
version		Sempre 1
storageDriverName	Nome do controlador de armazenamento	ontap-nas ontap-nas-economy, , ontap-nas-flexgroup ontap-san , , , ontap-san-economy
backendName	Nome personalizado ou back-end de storage	Nome do driver

Parâmetro	Descrição	Exemplo
managementLIF	Endereço IP de um cluster ou LIF de gerenciamento de SVM Um nome de domínio totalmente qualificado (FQDN) pode ser especificado. Pode ser definido para usar endereços IPv6 se o Astra Trident tiver sido instalado usando 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 aws campo, não precisará fornecer o managementLIF porque o Astra Trident recupera as informações do SVM managementLIF da AWS. Portanto, você deve fornecer credenciais para um usuário sob o SVM (por exemplo: Vsadmin) e o usuário deve ter a vsadmin função.	"10.0.0.1", "[2001:1234:abcd::fefe]"

Parâmetro	Descrição	Exemplo
dataLIF	Endereço IP do protocolo LIF. * ONTAP nas drivers*: Recomendamos especificar dataLIF. Se não for fornecido, o Astra Trident obtém LIFs de dados do SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) a ser usado para as operações de montagem NFS, permitindo que você crie um DNS de round-robin para balanceamento de carga em vários LIFs de dados. Pode ser alterado após a definição inicial. Consulte a . Drivers SAN ONTAP: Não especifique para iSCSI. O Astra Trident usa o mapa de LUN seletivo da ONTAP para descobrir as LIFs iSCSI necessárias para estabelecer uma sessão de vários caminhos. Um aviso é gerado se o dataLIF for definido explicitamente. Pode ser definido para usar endereços IPv6 se o Astra Trident tiver sido instalado usando 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 da política de exportação [Boolean]. Com autoExportPolicy as opções e autoExportCIDRs, o Astra Trident pode gerenciar políticas de exportação automaticamente.	false
autoExportCIDRs	Lista de CIDR para filtrar IPs de nós do Kubernetes quando autoExportPolicy está ativado. Com autoExportPolicy as opções e autoExportCIDRs, o Astra Trident pode gerenciar políticas de exportação automaticamente.	"["0.0.0.0/0", ":::/0"]"
labels	Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes	""

Parâmetro	Descrição	Exemplo
clientCertificate	Valor codificado em base64 do certificado do cliente. Usado para autenticação baseada em certificado	""
clientPrivateKey	Valor codificado em base64 da chave privada do cliente. Usado para autenticação baseada em certificado	""
trustedCACertificate	Valor codificado em base64 do certificado CA confiável. Opcional. Usado para autenticação baseada em certificado.	""
username	Nome de usuário para se conectar ao cluster ou SVM. Usado para autenticação baseada em credenciais. Por exemplo, vsadmin.	
password	Senha para se conectar ao cluster ou SVM. Usado para autenticação baseada em credenciais.	
svm	Máquina virtual de armazenamento para usar	Derivado se um SVM managementLIF for especificado.
storagePrefix	Prefixo usado ao provisionar novos volumes na SVM. Não pode ser modificado após a criação. Para atualizar esse parâmetro, você precisará criar um novo backend.	trident
limitAggregateUsage	Não especifique para o Amazon FSX for NetApp ONTAP. O fornecido fsxadmin e vsadmin não contém as permissões necessárias para recuperar o uso agregado e limitá-lo usando o Astra Trident.	Não utilizar.
limitVolumeSize	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse 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.	"" (não aplicado por padrão)
lunsPerFlexvol	O máximo de LUNs por FlexVol tem de estar no intervalo [50, 200]. Apenas SAN.	"100"

Parâmetro	Descrição	Exemplo
debugTraceFlags	Debug flags para usar ao solucionar problemas. Por exemplo, não use debugTraceFlags a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	nulo
nfsMountOptions	Lista separada por vírgulas de opções de montagem NFS. As opções de montagem para volumes persistentes do Kubernetes normalmente são especificadas em classes de storage, mas se nenhuma opção de montagem for especificada em uma classe de storage, o Astra Trident voltará a usar as opções de montagem especificadas no arquivo de configuração do back-end de storage. Se nenhuma opção de montagem for especificada na classe de storage ou no arquivo de configuração, o Astra Trident não definirá nenhuma opção de montagem em um volume persistente associado.	""
nasType	Configurar a criação de volumes NFS ou SMB. As opções são <code>nfs</code> , <code>smb</code> , ou <code>null</code> . Deve definir como <code>smb</code> para volumes SMB. A configuração como <code>null</code> padrão para volumes NFS.	<code>nfs</code>
qtreesPerFlexvol	Qtrees máximos por FlexVol, têm de estar no intervalo [50, 300]	"200"
smbShare	Você pode especificar uma das seguintes opções: O nome de um compartilhamento SMB criado usando o Console de Gerenciamento da Microsoft ou a CLI do ONTAP ou um nome para permitir que o Astra Trident crie o compartilhamento SMB. Esse parâmetro é necessário para backends do Amazon FSX for ONTAP.	<code>smb-share</code>

Parâmetro	Descrição	Exemplo
useREST	Parâmetro booleano para usar APIs REST do ONTAP. A visualização técnica useREST é fornecida como uma prévia técnica que é recomendada para ambientes de teste e não para cargas de trabalho de produção. Quando definido como <code>true</code> , o Astra Trident usará as APIs REST do ONTAP para se comunicar com o back-end. Esse recurso requer o ONTAP 9.11,1 e posterior. Além disso, a função de login do ONTAP usada deve ter acesso ao <code>ontap</code> aplicativo. Isso é satisfeito com as funções <code>cluster-admin</code> e <code>vsadmin</code> predefinidas.	false
aws	Você pode especificar o seguinte no arquivo de configuração do AWS FSX for ONTAP: - <code>fsxFilesystemID</code> : Especifique o ID do sistema de arquivos AWS FSX. <code>apiRegion</code> : Nome da região da API AWS. <code>apikey</code> : Chave da API da AWS. <code>secretKey</code> : Chave secreta da AWS.	"" "" ""
credentials	Especifique as credenciais do FSX SVM para armazenar no AWS Secret Manager. <code>name</code> : Nome do recurso Amazon (ARN) do segredo, que contém as credenciais do SVM. <code>type</code> : Defina para <code>awsarn</code> . "Crie um segredo do AWS Secrets Manager" Consulte para obter mais informações.	

Opções de configuração de back-end para volumes de provisionamento

Você pode controlar o provisionamento padrão usando essas opções na `defaults` seção da configuração. Para obter um exemplo, consulte os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
spaceAllocation	Alocação de espaço para LUNs	true
spaceReserve	Modo de reserva de espaço; "nenhum" (fino) ou "volume" (grosso)	none
snapshotPolicy	Política de instantâneos a utilizar	none

Parâmetro	Descrição	Padrão
qosPolicy	Grupo de políticas de QoS a atribuir aos volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento ou backend. O uso de grupos de política de QoS com o Astra Trident requer o ONTAP 9.8 ou posterior. Recomendamos o uso de um grupo de políticas de QoS não compartilhado e garantir que o grupo de políticas seja aplicado individualmente a cada componente. Um grupo de política de QoS compartilhado aplicará o limite máximo da taxa de transferência total de todos os workloads.	""
adaptiveQosPolicy	Grupo de políticas de QoS adaptável a atribuir para volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento ou backend. Não suportado pela ONTAP-nas-Economy.	""
snapshotReserve	Porcentagem de volume reservado para snapshots "0"	Se snapshotPolicy for none, else ""
splitOnClone	Divida um clone de seu pai na criação	false
encryption	Ative a criptografia de volume do NetApp (NVE) no novo volume; o padrão é false. O NVE deve ser licenciado e habilitado no cluster para usar essa opção. Se o NAE estiver ativado no back-end, qualquer volume provisionado no Astra Trident será o NAE ativado. Para obter mais informações, consulte: "Como o Astra Trident funciona com NVE e NAE" .	false
luksEncryption	Ativar encriptação LUKS. "Usar a configuração de chave unificada do Linux (LUKS)" Consulte a . Apenas SAN.	""
tieringPolicy	Política de disposição em camadas para usar none	snapshot-only Para configuração pré-ONTAP 9.5 SVM-DR

Parâmetro	Descrição	Padrão
unixPermissions	Modo para novos volumes. Deixe vazio para volumes SMB.	""
securityStyle	Estilo de segurança para novos volumes. Estilos de segurança e unix suporte de NFS mixed. Suporta SMB mixed e ntfs estilos de segurança.	O padrão NFS é unix. O padrão SMB é ntfs.

Prepare-se para provisionar volumes SMB

Você pode provisionar volumes SMB usando `ontap-nas` o driver. Antes de concluir [Integração de driver SAN e nas ONTAP](#) as etapas a seguir.

Antes de começar

Antes de provisionar volumes SMB usando `ontap-nas` o driver, você deve ter o seguinte:

- Um cluster do Kubernetes com um nó de controlador Linux e pelo menos um nó de trabalho do Windows que executa o Windows Server 2019. O Astra Trident é compatível com volumes SMB montados em pods executados apenas em nós do Windows.
- Pelo menos um segredo do Astra Trident que contém suas credenciais do active Directory. Para gerar segredo `smbcreds`:

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

- Um proxy CSI configurado como um serviço Windows. Para configurar um `csi-proxy`, "[GitHub: CSI Proxy](#)" consulte ou "[GitHub: CSI Proxy para Windows](#)" para nós do Kubernetes executados no Windows.

Passos

1. Criar compartilhamentos SMB. Você pode criar os compartilhamentos de administração SMB de duas maneiras usando o "[Microsoft Management Console](#)" snap-in pastas compartilhadas ou usando a CLI do ONTAP. Para criar compartilhamentos SMB usando a CLI do ONTAP:

- a. Se necessário, crie a estrutura do caminho do diretório para o compartilhamento.

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

- b. Crie um compartilhamento SMB associado ao SVM especificado:

```
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
```



"[Crie um compartilhamento SMB](#)" Consulte para obter detalhes completos.

2. Ao criar o back-end, você deve configurar o seguinte para especificar volumes SMB. Para obter todas as opções de configuração de back-end do FSX for ONTAP, "[Opções e exemplos de configuração do FSX for ONTAP](#)" consulte .

Parâmetro	Descrição	Exemplo
smbShare	Você pode especificar uma das seguintes opções: O nome de um compartilhamento SMB criado usando o Console de Gerenciamento da Microsoft ou a CLI do ONTAP ou um nome para permitir que o Astra Trident crie o compartilhamento SMB. Esse parâmetro é necessário para backends do Amazon FSX for ONTAP.	smb-share
nasType	Tem de estar definido para smb. Se nulo, o padrão é <code>nfs</code> .	smb
securityStyle	Estilo de segurança para novos volumes. Deve ser definido como ntfs ou mixed para volumes SMB.	ntfs Ou mixed para volumes SMB
unixPermissions	Modo para novos volumes. Deve ser deixado vazio para volumes SMB.	""

Configurar uma classe de armazenamento e PVC

Configure um objeto Kubernetes StorageClass e crie a classe de storage para instruir o Astra Trident a provisionar volumes. Crie um Persistentvolume (PV) e um PersistentVolumeClaim (PVC) que use o Kubernetes StorageClass configurado para solicitar acesso ao PV. Em seguida, pode montar o PV num pod.

Crie uma classe de armazenamento

Configurar um objeto Kubernetes StorageClass

O "[Objeto Kubernetes StorageClass](#)" identifica o Astra Trident como o provisionador que é usado para essa classe instrui o Astra Trident a provisionar um volume. Por exemplo:


```
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"
```

["Objetos Kubernetes e Trident"](#) Consulte para obter detalhes sobre como as classes de storage interagem com os PersistentVolumeClaim parâmetros e para controlar como o Astra Trident provisiona volumes.

Crie uma classe de armazenamento

Passos

1. Esse é um objeto do Kubernetes, então use `kubectl` para criá-lo no Kubernetes.

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

2. Agora você deve ver uma classe de storage **Basic-csi** no Kubernetes e Astra Trident, e o Astra Trident deve ter descoberto os pools no back-end.

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

Crie o PV e o PVC

A ["Persistentvolume"](#) (PV) é um recurso de armazenamento físico provisionado pelo administrador de cluster em um cluster do Kubernetes. O ["PersistentVolumeClaim"](#) (PVC) é um pedido de acesso ao Persistentvolume no cluster.

O PVC pode ser configurado para solicitar o armazenamento de um determinado tamanho ou modo de acesso. Usando o StorageClass associado, o administrador do cluster pode controlar mais do que o Persistentvolume e o modo de acesso, como desempenho ou nível de serviço.

Depois de criar o PV e o PVC, você pode montar o volume em um pod.

Manifestos de amostra

Persistentvolume Sample MANIFEST

Este manifesto de exemplo mostra um PV básico de 10Gi que está associado ao StorageClass . basic-csi

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

PersistentVolumeClaim amostra manifestos

Estes exemplos mostram opções básicas de configuração de PVC.

PVC com acesso RWO

Este exemplo mostra um PVC básico com acesso RWX associado a um StorageClass `basic-csi` chamado .

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

PVC com NVMe/TCP

Este exemplo mostra um PVC básico para NVMe/TCP com acesso RWO associado a um StorageClass `protection-gold` chamado .

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

Crie o PV e o PVC

Passos

1. Crie o PV.

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

2. Verifique o estado do PV.

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

3. Crie o PVC.

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

4. Verifique o estado do PVC.

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

"Objetos Kubernetes e Trident" Consulte para obter detalhes sobre como as classes de storage interagem com os PersistentVolumeClaim parâmetros e para controlar como o Astra Trident provisiona volumes.

Atributos do Astra Trident

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

Atributo	Tipo	Valores	Oferta	Pedido	Suportado por
1	cadeia de caracteres	hdd, híbrido, ssd	Pool contém Mídia desse tipo; híbrido significa ambos	Tipo de material especificado	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san, SolidFire-san
ProvisioningType	cadeia de caracteres	fino, grosso	O pool é compatível com esse método de provisionamento	Método de provisionamento especificado	thick: all ONTAP; thin: all ONTAP & SolidFire-san

Atributo	Tipo	Valores	Oferta	Pedido	Suportado por
BackendType	cadeia de caracteres	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san, SolidFire-san, gcp-cvs, azure-NetApp-files, ONTAP-san-economy	Pool pertence a este tipo de backend	Back-end especificado	Todos os drivers
instantâneos	bool	verdadeiro, falso	O pool é compatível com volumes com snapshots	Volume com instantâneos ativados	ONTAP-nas, ONTAP-san, SolidFire-san, gcp-cvs
clones	bool	verdadeiro, falso	O pool é compatível com volumes de clonagem	Volume com clones ativados	ONTAP-nas, ONTAP-san, SolidFire-san, gcp-cvs
criptografia	bool	verdadeiro, falso	O pool é compatível com volumes criptografados	Volume com encriptação ativada	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-flexgroups, ONTAP-san
IOPS	int	número inteiro positivo	O pool é capaz de garantir IOPS nessa faixa	Volume garantido estas operações de entrada/saída por segundo	SolidFire-san

1: Não suportado pelos sistemas ONTAP Select

Implantar um aplicativo de amostra

Implantar um aplicativo de amostra.

Passos

1. Monte o volume num pod.

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

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: 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

```



Pode monitorizar o progresso utilizando `kubectl get pod --watch`o .

2. Verifique se o volume está montado no `/my/mount/path`.

```
kubectl exec -it task-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	

1. Agora você pode excluir o Pod. O aplicativo Pod não existirá mais, mas o volume permanecerá.

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

Configure o complemento Astra Trident EKS em um cluster EKS

O Astra Trident simplifica o gerenciamento de armazenamento do Amazon FSX for NetApp ONTAP no Kubernetes para permitir que seus desenvolvedores e administradores se concentrem na implantação de aplicações. O complemento Astra Trident EKS inclui os patches de segurança mais recentes, correções de bugs e é validado pela AWS para funcionar com o Amazon EKS. O complemento EKS permite

que você garanta consistentemente que seus clusters do Amazon EKS estejam seguros e estáveis e reduza a quantidade de trabalho que você precisa fazer para instalar, configurar e atualizar complementos.

Pré-requisitos

Antes de configurar o complemento Astra Trident para AWS EKS, verifique se você tem o seguinte:

- Uma conta de cluster do Amazon EKS com assinatura complementar
- Permissões da AWS para o marketplace da AWS:
"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 existente do Amazon FSX for NetApp ONTAP

Passos

1. No cluster do EKS Kubernetes, navegue até a guia **Complementos**.

The screenshot shows the AWS EKS console interface for a cluster named 'tri-env-eks'. At the top, there are buttons for 'Delete cluster' and 'Upgrade version'. Below this is a notification bar about the end of standard support for Kubernetes version 1.30. The main section is titled 'Cluster info' and displays details such as Status (Active), Kubernetes version (1.30), Support period (Standard support until July 28, 2025), and Provider (EKS). A navigation bar below the cluster info shows various tabs: Overview, Resources, Compute, Networking, Add-ons (selected and marked with a '1'), Access, Observability, Upgrade insights, Update history, and Tags. Below the navigation bar is another notification bar stating 'New versions are available for 3 add-ons.' The 'Add-ons (3)' section is visible, showing a search bar, filters for 'Any category' and 'Any status', and a 'Get more add-ons' button. The bottom of the section shows '3 matches' and a pagination control.

2. Vá para **Complementos do AWS Marketplace** e escolha a categoria *storage*.

AWS Marketplace add-ons (1)

Discover, subscribe to and configure EKS add-ons to enhance your EKS clusters.

Filtering options

Any category ▼
NetApp, Inc. ▼
Any pricing model ▼
Clear filters

NetApp, Inc. ✕
1

NetApp Trident
☐

NetApp Trident streamlines Amazon FSx for NetApp ONTAP storage management in Kubernetes to let your developers and administrators focus on application deployment. FSx for ONTAP flexibility, scalability, and integration capabilities make it the ideal choice for organizations seeking efficient containerized storage workflows. [Product details](#)

Standard Contract

Category	Listed by	Supported versions	Pricing starting at
storage	NetApp, Inc.	1.30, 1.29, 1.28, 1.27, 1.26, 1.25, 1.24, 1.23	View pricing details


Cancel
Next

3. Localize **NetApp Trident** e marque a caixa de seleção do complemento Astra Trident.
4. Escolha a versão desejada do complemento.

NetApp Trident

Remove add-on

Listed by NetApp	Category storage	Status ✓ Ready to install
----------------------------	---------------------	------------------------------

 **You're subscribed to this software**
You can view the terms and pricing details for this product or choose another offer if one is available.

View subscription


×

Version
Select the version for this add-on.

v24.6.1-eksbuild.1

Select IAM role
Select an IAM role to use with this add-on. To create a new custom role, follow the instructions in the [Amazon EKS User Guide](#).

Not set



► Optional configuration settings

Cancel

Previous

Next

5. Selecione a opção função do IAM para herdar do nó.

Review and add

Step 1: Select add-ons

[Edit](#)

Selected add-ons (1)

< 1 >

Add-on name



Type



Status

netapp_trident-operator

storage

✓ Ready to install

Step 2: Configure selected add-ons settings

[Edit](#)

Selected add-ons version (1)

< 1 >

Add-on name



Version



IAM role for service account (IRSA)

netapp_trident-operator

v24.6.1-eksbuild.1

Not set

[Cancel](#)[Previous](#)[Create](#)

6. (Opcional) Configure quaisquer configurações opcionais conforme necessário e selecione **Next**.

Siga o esquema de configuração **Add-on** e defina o parâmetro `configurationValues` na seção **valores de configuração** para o Role-arn criado na etapa anterior (o valor deve estar no seguinte formato:

`eks.amazonaws.com/role-arn:`

`arn:aws:iam::464262061435:role/AmazonEKS_FSXN_CSI_DriverRole`). Se você selecionar **Substituir** para o método de resolução de conflitos, uma ou mais configurações para o suplemento existente podem ser sobrescritas com as configurações de complemento do Amazon EKS. Se você não ativar essa 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 essa opção, certifique-se de que o complemento do Amazon EKS não gerencie as configurações que você precisa para gerenciar automaticamente.



Ao configurar o parâmetro opcional `cloudIdentity`, certifique-se de especificar `AWS` como o `cloudProvider` durante a instalação do Trident usando o complemento EKS.

Select IAM role
Select an IAM role to use with this add-on. To create a new custom role, follow the instructions in the [Amazon EKS User Guide](#).

Not set

▼ Optional configuration settings

Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.

```
{
  "$id": "http://example.com/example.json",
  "$schema": "https://json-schema.org/draft/2019-09/schema",
  "default": {},
  "examples": [
    {
      "cloudIdentity": ""
    }
  ],
  "properties": {
    "cloudIdentity": {
      "default": "",
      "examples": [

```

Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.

```
1 {
2   "cloudIdentity": "'eks.amazonaws.com/role-arn: arn:aws
3   :iam::139763910815:role
4   /AmazonEKS_FSXN_CSI_DriverRole'",
  "cloudProvider": "AWS"
}
```

7. Selecione **criar**.
8. Verifique se o status do complemento é *ativo*.

Add-ons (1) Info

View details Edit Remove Get more add-ons

netapp Any category Any status 1 match < 1 >

NetApp Astra Trident by NetApp

Astra Trident streamlines Amazon FSx for NetApp ONTAP storage management in Kubernetes to let your developers and administrators focus on application deployment. FSx for ONTAP flexibility, scalability, and integration capabilities make it the ideal choice for organizations seeking efficient containerized storage workflows. [Product details](#)

Category	Status	Version	IAM role for service account (IRSA)	Listed by
storage	Active	v24.6.1-eksbuild.1	Not set	NetApp, Inc.

View subscription

Instale/desinstale o complemento Astra Trident EKS usando a CLI

Instale o complemento Astra Trident EKS usando a CLI:

O seguinte exemplo de comando instala o complemento Astra Trident EKS:

```
eksctl create addon --cluster K8s-arm --name netapp_trident-operator --version v24.6.1-eksbuild
```

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v24.6.1-eksbuild.1 (Com uma versão dedicada)
```



Ao configurar o parâmetro opcional `cloudIdentity`, certifique-se de especificar `cloudProvider` durante a instalação do Trident usando o complemento EKS.

Desinstale o complemento Astra Trident EKS usando a CLI:

O comando a seguir desinstala o complemento Astra Trident EKS:

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

Crie backends com kubectl

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

`TridentBackendConfig`

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

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

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

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



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

Veja o exemplo a seguir para o formato do `TridentBackendConfig` CR:

```

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

```

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

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

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

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



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



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

Visão geral dos passos

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

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

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

Etapa 1: Crie um segredo do Kubernetes

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

```
kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: password
```

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

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

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

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

Passo 2: Crie o TridentBackendConfig CR

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

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

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

Etapa 3: Verifique o status do TridentBackendConfig CR

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

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

NAME	BACKEND NAME	BACKEND UUID
backend-tbc-ontap-san	ontap-san-backend	8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
Bound	Success	

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

A fase pode ter um dos seguintes valores:

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

CR.

- Se um ou mais PVCs estiverem presentes no back-end, ele vai para um estado de exclusão. Posteriormente, o `TridentBackendConfig` CR entra também na fase de eliminação. O back-end e `TridentBackendConfig` são excluídos somente depois que todos os PVCs são excluídos.
- **Lost:** O back-end associado ao `TridentBackendConfig` CR foi acidentalmente ou deliberadamente excluído e o `TridentBackendConfig` CR ainda tem uma referência ao back-end excluído. O `TridentBackendConfig` CR ainda pode ser eliminado independentemente do `deletionPolicy` valor.
- **Unknown:** O Astra Trident não consegue determinar o estado ou a existência do back-end associado ao `TridentBackendConfig` CR. Por exemplo, se o servidor de API não estiver respondendo ou se o `tridentbackends.trident.netapp.io` CRD estiver ausente. Isso pode exigir intervenção.

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

(Opcional) passo 4: Obtenha mais detalhes

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

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

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

Além disso, você também pode obter um despejo YAML/JSON do `TridentBackendConfig`.

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

```

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

```

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

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



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

Gerenciar backends

Execute o gerenciamento de back-end com o kubectl

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

Excluir um back-end

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

Execute o seguinte comando:

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

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

Veja os backends existentes

Execute o seguinte comando:

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

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

Atualize um back-end

Pode haver várias razões para atualizar um backend:

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

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

- Os parâmetros (como o nome do SVM do ONTAP sendo usado) precisam ser atualizados.
 - 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
```



- Se uma atualização de back-end falhar, o back-end continuará em sua última configuração conhecida. Pode visualizar os registros para determinar a causa executando `kubectl get tbc <tbc-name> -o yaml -n trident` ou `kubectl describe tbc <tbc-name> -n trident`.
- Depois de identificar e corrigir o problema com o arquivo de configuração, você pode executar novamente o comando `update`.

Execute o gerenciamento de back-end com o `tridentctl`

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

Crie um backend

Depois de criar um "[arquivo de configuração de back-end](#)", execute o seguinte comando:

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

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

```
tridentctl logs -n trident
```

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

Excluir um back-end

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

1. Recuperar o nome do backend:

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

2. Excluir o backend:

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



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

Veja os backends existentes

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

- Para obter um resumo, execute o seguinte comando:

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

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

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

Atualize um back-end

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

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

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

```
tridentctl logs -n trident
```

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

Identificar as classes de armazenamento que usam um back-end

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

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

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

Alternar entre opções de gerenciamento de back-end

Saiba mais sobre as diferentes maneiras de gerenciar back-ends no Astra Trident.

Opções para gerenciar backends

Com a introdução `TridentBackendConfig`, os administradores agora têm duas maneiras exclusivas de gerenciar backends. Isso coloca as seguintes perguntas:

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

Gerenciar `tridentctl` backends usando `TridentBackendConfig`

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

Isso se aplicará aos seguintes cenários:

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

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

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

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

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

Passo 0: Identifique o backend

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

```
tridentctl get backend ontap-nas-backend -n trident
```

```

+-----+-----+
+-----+-----+-----+-----+
|           NAME           | STORAGE DRIVER |           UUID
| STATE   | VOLUMES |
+-----+-----+
+-----+-----+-----+-----+
| ontap-nas-backend      | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+-----+-----+
+-----+-----+-----+-----+

```

```
cat ontap-nas-backend.json
```

```

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

  "defaults": {
    "spaceReserve": "none",
    "encryption": "false"
  },
  "labels": {"store": "nas_store"},
  "region": "us_east_1",
  "storage": [
    {
      "labels": {"app": "msoffice", "cost": "100"},
      "zone": "us_east_1a",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "true",
        "unixPermissions": "0755"
      }
    },
    {
      "labels": {"app": "mysqldb", "cost": "25"},
      "zone": "us_east_1d",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "false",
        "unixPermissions": "0775"
      }
    }
  ]
}

```

```

    }
  }
]
}

```

Etapa 1: Crie um segredo do Kubernetes

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

```

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

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

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

```

Passo 2: Crie um TridentBackendConfig CR

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

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

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


```

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

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

```

Etapas 3: Verifique o status do TridentBackendConfig CR

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

```
kubectl get tbc tbc-ontap-nas-backend -n trident
```

NAME	BACKEND NAME	BACKEND UUID
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
ontap-nas-backend	ontap-nas	52f2eb10-e4c6-4160-99fc-96b3be5ab5d7
online	25	

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

Gerenciar TridentBackendConfig **backends usando** tridentctl

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

Passo 0: Identifique o backend

Por exemplo, vamos supor que o seguinte backend foi criado usando TridentBackendConfig:

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

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

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

Passo 1: Confirmar deletionPolicy está definido como retain

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

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

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

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



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

Etapa 2: Exclua o `TridentBackendConfig` CR

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

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

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

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

Criar e gerenciar classes de armazenamento

Crie uma classe de armazenamento

Configure um objeto Kubernetes `StorageClass` e crie a classe de storage para instruir o Astra Trident a provisionar volumes.

Configurar um objeto Kubernetes `StorageClass`

O "[Objeto Kubernetes StorageClass](#)" identifica o Astra Trident como o provisionador que é usado para essa classe e instrui o Astra Trident a provisionar um volume. Por exemplo:

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

"Objetos Kubernetes e Trident" Consulte para obter detalhes sobre como as classes de storage interagem com os PersistentVolumeClaim parâmetros e para controlar como o Astra Trident provisiona volumes.

Crie uma classe de armazenamento

Depois de criar o objeto StorageClass, você pode criar a classe de armazenamento. [Amostras da classe de armazenamento](#) fornece algumas amostras básicas que você pode usar ou modificar.

Passos

1. Esse é um objeto do Kubernetes, então use `kubectl` para criá-lo no Kubernetes.

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

2. Agora você deve ver uma classe de storage **Basic-csi** no Kubernetes e Astra Trident, e o Astra Trident deve ter descoberto os pools no back-end.

```

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"
        ]
      }
    }
  ]
}

```

Amostras da classe de armazenamento

O Astra Trident ["definições de classe de armazenamento simples para backends específicos"](#) fornece .

Alternativamente, você pode editar `sample-input/storage-class-csi.yaml.tmpl` o arquivo que vem com o instalador e substituir `BACKEND_TYPE` pelo nome do driver de armazenamento.

```
./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 exibir classes de armazenamento existentes, definir uma classe de armazenamento padrão, identificar o back-end da classe de armazenamento e excluir classes de armazenamento.

Exibir as classes de armazenamento existentes

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

```
kubectl get storageclass
```

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

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

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

```
tridentctl get storageclass
```

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

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

Defina uma classe de armazenamento padrão

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

- Defina uma classe de armazenamento padrão definindo a anotação `storageclass.kubernetes.io/is-default-class` como verdadeira na definição da classe de armazenamento. De acordo com a especificação, qualquer outro valor ou ausência da anotação é interpretado como falso.
- Você pode configurar uma classe de armazenamento existente para ser a classe de armazenamento padrão usando o seguinte comando:

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

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

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

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



Deve haver apenas uma classe de armazenamento padrão no cluster de cada vez. O Kubernetes não impede tecnicamente que você tenha mais de um, mas se comportará como se não houvesse nenhuma classe de storage padrão.

Identificar o back-end de uma classe de storage

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

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

Excluir uma classe de armazenamento

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

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


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

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



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

Provisionar e gerenciar volumes

Provisionar um volume

Crie um `PersistentVolume` (PV) e um `PersistentVolumeClaim` (PVC) que use o Kubernetes `StorageClass` configurado para solicitar acesso ao PV. Em seguida, pode montar o PV num pod.

Visão geral

A "[Persistentvolume](#)" (PV) é um recurso de armazenamento físico provisionado pelo administrador de cluster em um cluster do Kubernetes. O "[PersistentVolumeClaim](#)" (PVC) é um pedido de acesso ao `Persistentvolume` no cluster.

O PVC pode ser configurado para solicitar o armazenamento de um determinado tamanho ou modo de acesso. Usando o `StorageClass` associado, o administrador do cluster pode controlar mais do que o `Persistentvolume` e o modo de acesso, como desempenho ou nível de serviço.

Depois de criar o PV e o PVC, você pode montar o volume em um pod.

Manifestos de amostra

Persistentvolume Sample MANIFEST

Este manifesto de exemplo mostra um PV básico de 10Gi que está associado ao StorageClass . basic-csi

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

PersistentVolumeClaim amostra manifestos

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 `basic-csi` chamado .

```
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 associado a um StorageClass `protection-gold` chamado .

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

Amostras de manifesto 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: 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
```

Configuração básica NVMe/TCP

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

Crie o PV e o PVC

Passos

1. Crie o PV.

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

2. Verifique o estado do PV.

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

3. Crie o PVC.

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

4. Verifique o estado do PVC.

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

5. Monte o volume num pod.

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



Pode monitorizar o progresso utilizando `kubectl get pod --watch`.

6. Verifique se o volume está montado no /my/mount/path.

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

7. Agora você pode excluir o Pod. O aplicativo Pod não existirá mais, mas o volume permanecerá.

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

["Objetos Kubernetes e Trident"](#) Consulte para obter detalhes sobre como as classes de storage interagem com os PersistentVolumeClaim parâmetros e para controlar como o Astra Trident provisiona volumes.

Expanda volumes

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

Expanda um volume iSCSI

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



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

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

Edite a definição StorageClass para definir o `allowVolumeExpansion` campo 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.

Etapa 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 tamanho recém-desejado, que deve ser maior do 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
```

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

```
kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY
san-pvc	Bound	pvc-8a814d62-bd58-4253-b0d1-82f2885db671	1Gi

```
kubectl get pv
```

NAME	RECLAIM POLICY	STATUS	CLAIM	CAPACITY	ACCESS MODES	AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671	Delete	Bound	default/san-pvc	1Gi	RWO	10s

Passo 3: Defina um pod que prende o PVC

Fixe o PV a um pod para que ele seja redimensionado. Existem dois cenários ao redimensionar um iSCSI PV:

- Se o PV estiver conectado a um pod, o Astra Trident expande o volume no back-end de armazenamento, refaz o dispositivo e redimensiona o sistema de arquivos.
- Ao tentar redimensionar um PV não anexado, o Astra Trident expande o volume no back-end de armazenamento. Depois que o PVC é ligado a um pod, o Trident refaz o dispositivo e redimensiona o sistema de arquivos. Em seguida, o Kubernetes atualiza o tamanho do PVC após a operação de expansão ter sido concluída com sucesso.

Neste exemplo, é criado um pod 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
```


Passo 4: Expanda o PV

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

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

Etapa 5: Validar a expansão

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

```
kubectl get pvc san-pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound       pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi
RWO           ontap-san    11m

kubectl get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY STATUS    CLAIM          STORAGECLASS  REASON    AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi        RWO
Delete              Bound      default/san-pvc  ontap-san    12m

tridentctl get volumes -n trident
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
+-----+-----+-----+-----+
|          BACKEND UUID  | STATE | MANAGED |
+-----+-----+-----+-----+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san |
| block | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

Expandir um volume NFS

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

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

Para redimensionar um PV NFS, o administrador primeiro precisa configurar a classe de armazenamento para permitir a expansão de volume definindo o `allowVolumeExpansion` campo para `true`:

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

Se você já criou uma classe de armazenamento sem essa opção, você pode simplesmente editar a classe de armazenamento existente usando `kubectl edit storageclass` para permitir a expansão de volume.

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

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

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

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

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

Passo 3: Expanda o PV

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

```

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

```

Etapas 4: Validar a expansão

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

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

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

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

Importar volumes

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

Visão geral e considerações

Você pode importar um volume para o Astra Trident para:

- Containerize um aplicativo e reutilize seu conjunto de dados existente
- Use um clone de um conjunto de dados para uma aplicação efêmera
- Reconstruir um cluster do Kubernetes com falha
- Migrar dados da aplicação durante a recuperação de desastre

Considerações

Antes de importar um volume, reveja as seguintes considerações.

- O Astra Trident pode importar apenas volumes ONTAP do tipo RW (leitura-gravação). Os volumes do tipo DP (proteção de dados) são volumes de destino do SnapMirror. Você deve quebrar a relação espelhada antes de importar o volume para o Astra Trident.

- Sugerimos importar volumes sem conexões ativas. Para importar um volume usado ativamente, clonar o volume e, em seguida, executar a importação.



Isso é especialmente importante para volumes de bloco, já que o Kubernetes não sabia da 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, o Astra Trident não usa esse parâmetro durante a importação. As classes de armazenamento são usadas durante a criação de volume para selecionar 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 se o volume existir 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. Depois que o volume é importado pelo driver de armazenamento, o PV é criado com uma `ClaimRef` para o PVC.
 - A política de recuperação é inicialmente definida como `retain` no PV. Depois que o Kubernetes vincula com êxito o PVC e o PV, a política de recuperação é atualizada para corresponder à política de recuperação da Classe de armazenamento.
 - Se a política de recuperação da Classe de armazenamento for `delete`, o volume de armazenamento será excluído quando o PV for excluído.
- Por padrão, o Astra Trident gerencia o PVC e renomeia o `FlexVol` e o LUN no back-end. Você pode passar o `--no-manage` sinalizador para importar um volume não gerenciado. Se você usar `--no-manage`o` , o Astra Trident não realiza nenhuma operação adicional no PVC ou no 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 redimensionamento de volume também são ignoradas.



Essa opção é útil se você quiser usar o Kubernetes para workloads em contêineres, mas de outra forma quiser gerenciar o ciclo de vida do volume de storage fora do Kubernetes.

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

Importar um volume

Pode utilizar `tridentctl import` para importar um volume.

Passos

1. Crie o arquivo PVC (Persistent volume Claim) (por exemplo, `pvc.yaml`) que será usado para criar o PVC. O ficheiro PVC deve incluir `name namespace` , , `accessModes storageClassName` e . Opcionalmente, você pode especificar `unixPermissions` em sua definição de PVC.

O seguinte é um exemplo de uma especificação mínima:

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



Não inclua parâmetros adicionais, como nome PV ou tamanho do volume. Isso pode fazer com que o comando de importação falhe.

2. Use o `tridentctl import volume` comando para especificar o nome do back-end do Astra Trident que contém o volume e o nome que identifica exclusivamente o volume no storage (por exemplo: ONTAP FlexVol, Element volume, caminho Cloud Volumes Service). O `-f` argumento é necessário para especificar o caminho para o arquivo PVC.

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

Exemplos

Reveja os exemplos de importação de volume a seguir para drivers suportados.

ONTAP nas e ONTAP nas FlexGroup

O Astra Trident é compatível com a importação de volume usando `ontap-nas` os drivers e `ontap-nas-flexgroup`



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

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

Exemplos de ONTAP nas

A seguir mostra um exemplo de um volume gerenciado e uma importação de volume não gerenciado.

Volume gerenciado

O exemplo a seguir importa um volume nomeado `managed_volume` em um backend chamado `ontap_nas`:

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

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

Volume não gerenciado

Ao usar o `--no-manage` argumento, o Astra Trident não renomeará o volume.

O exemplo a seguir é importado `unmanaged_volume` `ontap_nas` no backend:

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

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

San ONTAP

O Astra Trident é compatível com a importação de volume usando `ontap-san` o driver. A importação de volume não é suportada usando `ontap-san-economy` o driver.

O Astra Trident pode importar ONTAP SAN FlexVols que contenham um único LUN. Isso é consistente com o `ontap-san` driver, que cria um FlexVol para cada PVC e um LUN dentro do FlexVol. O Astra Trident importa o FlexVol e associa-o à definição de PVC.

A seguir mostra um exemplo de um volume gerenciado e uma importação de volume não gerenciado.

O exemplo a seguir importa `ontap-san-managed` o FlexVol que está presente no `ontap_san_default` back-end:

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

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

NAME	SIZE	STORAGE CLASS
pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228	1.0 GiB	san-blog
block	online	false

175

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

Elemento

O Astra Trident é compatível com o software NetApp Element e a importação de volume NetApp HCI usando `solidfire-san` o driver.



O driver Element suporta nomes de volume duplicados. No entanto, o Astra 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.

Exemplo de elemento

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

Google Cloud Platform

O Astra Trident é compatível com a importação de volume usando `gcp-cvs` o driver.



Para importar um volume com o suporte do NetApp Cloud Volumes Service no Google Cloud Platform, identifique o volume pelo caminho de volume. O caminho do volume é a parte do caminho de exportação do volume após o `:/`. Por exemplo, se o caminho de exportação for `10.0.0.1:/adroit-jolly-swift`, o caminho do volume será `adroit-jolly-swift`.

Exemplo do Google Cloud Platform

O exemplo a seguir importa um gcp-cvs volume no back-end gcpcvs_YEppr com o caminho de volume adroit-jolly-swift do .

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

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

Azure NetApp Files

O Astra Trident é compatível com a importação de volume usando azure-netapp-files o driver.



Para importar um volume 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 será importvol1.

Exemplo de Azure NetApp Files

O exemplo a seguir importa um azure-netapp-files volume no back-end azurenetaappfiles_40517 com o caminho do volume importvol1 .

```
tridentctl import volume azurenetaappfiles_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	

Personalizar nomes e rótulos de volume

Com o Astra Trident, você pode atribuir nomes e rótulos significativos aos volumes criados. Isso ajuda a identificar e mapear facilmente volumes para seus respectivos recursos do Kubernetes (PVCs). Você também pode definir modelos no nível de back-end para criar nomes de volume personalizados e rótulos personalizados; todos os volumes que você criar, importar ou clonar aderirão aos modelos.

Antes de começar

Nomes de volume e etiquetas personalizáveis suportam:

1. Operações de criação, importação e clonagem de volume.
2. No caso do driver ONTAP-nas-Economy, apenas o nome do volume Qtree está em conformidade com o modelo de nome.
3. No caso do driver ONTAP-san-Economy, apenas o nome LUN está em conformidade com o modelo de nome.

Limitações

1. Os nomes de volume personalizáveis são compatíveis apenas com drivers locais da ONTAP.
2. Nomes de volume personalizáveis não se aplicam a volumes existentes.

Principais comportamentos de nomes de volume personalizáveis

1. Se ocorrer uma falha devido à sintaxe inválida em um modelo de nome, a criação do backend falhará. No entanto, se o aplicativo modelo falhar, o volume será nomeado de acordo com a convenção de nomenclatura existente.
2. O prefixo de armazenamento não é aplicável quando um volume é nomeado usando um modelo de nome da configuração de back-end. Qualquer valor de prefixo desejado pode ser adicionado diretamente ao modelo.

Exemplos de configuração de backend com modelo de nome e rótulos

Modelos de nome personalizados podem ser definidos no nível raiz e/ou pool.

Exemplo de nível de 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.RequestName}}"
  },
  "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 modelo 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 das importações de volume, as etiquetas são atualizadas apenas se o volume existente tiver etiquetas num formato específico. Por exemplo `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}:.`
2. No caso de importações de volume gerenciado, o nome do volume segue o modelo de nome definido no nível raiz na definição de back-end.
3. O Astra Trident não oferece suporte ao uso de um operador de fatia com o prefixo de storage.
4. Se os modelos não resultarem em nomes de volume exclusivos, o Astra Trident anexará alguns caracteres aleatórios para criar nomes de volume exclusivos.
5. Se o nome personalizado para um volume de economia nas exceder 64 caracteres de comprimento, o Astra 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 nome, o processo de criação de volume falhará.

Compartilhar um volume NFS entre namespaces

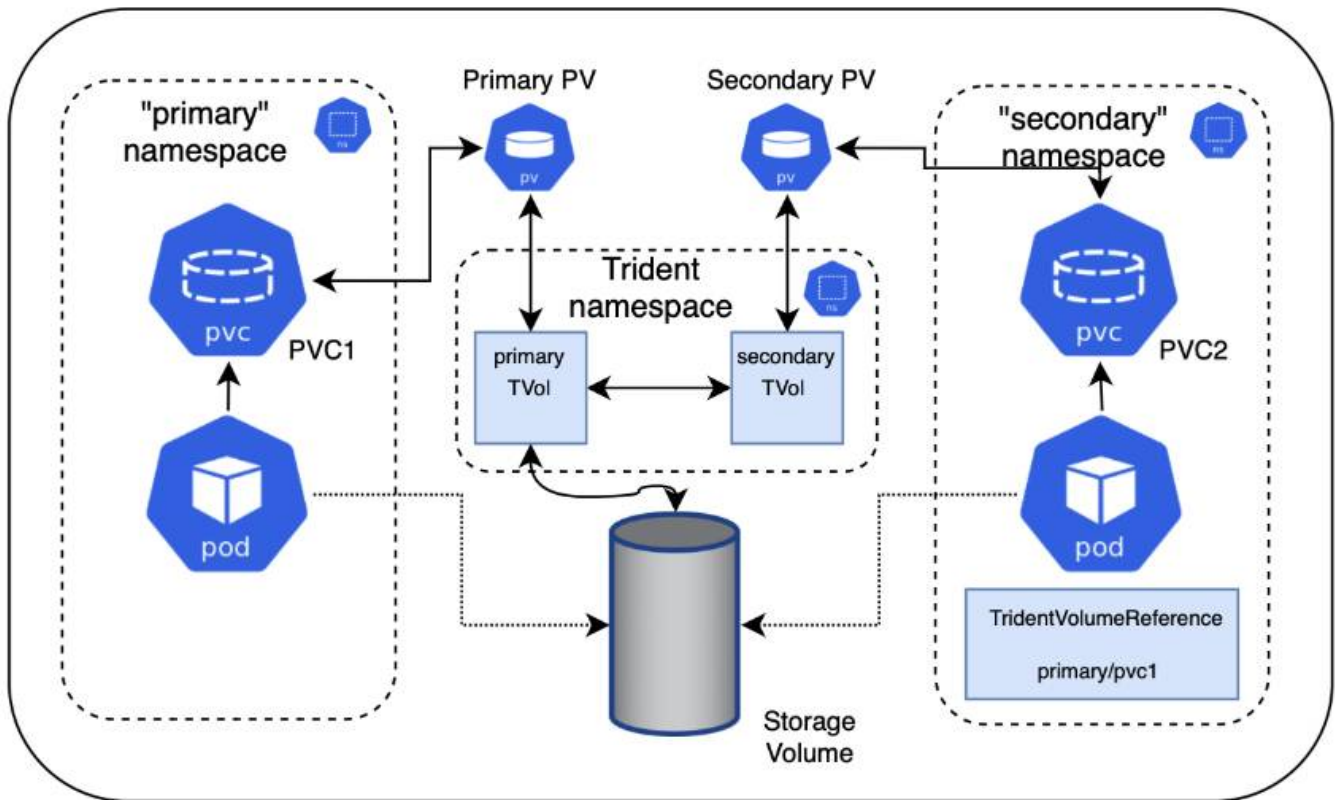
Com o Astra Trident, você pode criar um volume em um namespace principal e compartilhá-lo em um ou mais namespaces secundários.

Caraterísticas

O Astra TridentVolumeReference CR permite que você compartilhe com segurança volumes NFS ReadWriteMany (RWX) em um ou mais namespaces do Kubernetes. Essa solução nativa do Kubernetes tem os seguintes benefícios:

- Vários níveis de controle de acesso para garantir a segurança
- Funciona com todos os drivers de volume Trident NFS
- Não há dependência do tridentctl ou de qualquer outro recurso do Kubernetes não nativo

Este diagrama ilustra o compartilhamento de volumes NFS em dois namespaces do Kubernetes.



Início rápido

Você pode configurar o compartilhamento de volume NFS em apenas algumas etapas.

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 **CredentVolumeReference** CR.

3

Crie **TridentVolumeReference** no namespace de destino

O proprietário do namespace de destino cria o **TridentVolumeReference** CR para se referir 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 namespace requer colaboração e ação do proprietário

do namespace de origem, do administrador do cluster e do proprietário do namespace de destino. A função de 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 `shareToNamespace` anotação.

```
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
```

O Astra Trident cria o PV e o volume de storage NFS no back-end.



- Você pode compartilhar o PVC para vários namespaces usando uma lista delimitada 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. **Cluster admin:** Crie a função personalizada e kubeconfig para conceder permissão ao proprietário do namespace de destino para criar o `TridentVolumeReference` CR no namespace de destino.
3. **Proprietário do namespace de destino:** Crie um `CredentVolumeReference` CR no namespace de destino que se refere 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 `shareFromPVC` anotação 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

```



O tamanho do PVC de destino deve ser inferior ou igual ao PVC de origem.

Resultados

O Astra 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 PV de origem. O PVC e o PV de destino aparecem encadernados normalmente.

Eliminar um volume compartilhado

Você pode excluir um volume compartilhado entre vários namespaces. O Astra Trident removerá o acesso ao volume no namespace de origem e manterá acesso para outros namespaces que compartilham o volume. Quando todos os namespaces que fazem referência ao volume são removidos, o Astra Trident exclui o volume.

`tridentctl get` Use para consultar volumes subordinados

Usando o `tridentctl` utilitário, você pode executar o `get` comando para obter volumes subordinados. Para obter mais informações, consulte o `tridentctl` [comandos e opções](#).

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

Bandeiras -

- `-h, --help`: Ajuda para volumes.
- `--parentOfSubordinate string`: Limitar consulta ao volume de origem subordinado.
- `--subordinateOf string`: Limitar consulta a subordinados de volume.

Limitações

- O Astra Trident não pode impedir que namespaces de destino gravem no volume compartilhado. Você deve usar o bloqueio de arquivos ou outros processos para evitar a substituição de dados de volume compartilhado.
- Não é possível revogar o acesso ao PVC de origem removendo as `shareToNamespace` anotações ou `shareFromNamespace` excluindo o `TridentVolumeReference` CR. Para revogar o acesso, você deve excluir o PVC subordinado.
- Snapshots, clones e espelhamento não são possíveis em volumes subordinados.

Para mais informações

Para saber mais sobre o acesso ao volume entre namespace:

- ["Compartilhamento de volumes entre namespaces: Diga olá ao acesso ao volume entre namespace"](#) Visite [.NetAppTV](#).
- Assista à demonstração no ["NetAppTV"](#).

Replique volumes usando o SnapMirror

Com o Astra Control Provisioner, você pode criar relacionamentos de espelhamento entre um volume de origem em um cluster e o volume de destino no cluster peered para replicação de dados para recuperação de desastres. Você pode usar uma Definição de recursos personalizados (CRD) para executar as seguintes operações:

- Criar relações de espelhamento entre volumes (PVCs)
- Remova as relações de espelho entre volumes
- Quebre as relações do espelho
- Promover o volume secundário durante as condições de desastre (failovers)
- Realizar a transição sem perda de aplicativos do cluster para o cluster (durante failovers planejados ou migrações)

Pré-requisitos de replicação

Certifique-se de que os seguintes pré-requisitos sejam atendidos antes de começar:

Clusters de ONTAP

- **Provisioner:** O Astra Control Provisioner versão 23,10 ou posterior deve existir nos clusters do Kubernetes de origem e destino que utilizam o ONTAP como um back-end.
- **Licenças:** As licenças assíncronas do ONTAP SnapMirror usando o pacote proteção de dados devem estar ativadas nos clusters ONTAP de origem e destino. ["Visão geral do licenciamento do SnapMirror no ONTAP"](#) Consulte para obter mais informações.

Peering

- **Cluster e SVM:** Os backends de storage do ONTAP devem ser colocados em Contato. ["Visão geral do peering de cluster e SVM"](#) Consulte para obter mais informações.



Certifique-se de que os nomes do SVM usados na relação de replicação entre dois clusters ONTAP sejam exclusivos.

- **Astra Control Provisioner e SVM:** Os SVMs remotas com peering devem estar disponíveis para o Astra Control Provisioner no cluster de destino.

Drivers suportados

- A replicação de volume é compatível com os drivers ONTAP-nas e ONTAP-san.

Crie um PVC espelhado

Siga estas etapas e use os exemplos CRD para criar relação de espelhamento entre volumes primário e secundário.

Passos

1. Execute as etapas a seguir no cluster primário do Kubernetes:
 - a. Crie um objeto StorageClass com o `trident.netapp.io/replication: true` parâmetro.

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 criado anteriormente.

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
```

O Astra Control Provisioner obtém as informações internas do volume e do estado atual de proteção de dados (DP) do volume e, em seguida, preenche o campo de status do MirrorRelationship.

- d. Obtenha o tridentMirrorRelationship CR para obter o nome interno e SVM do PVC.

```
kubect1 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 etapas a seguir no cluster secundário do Kubernetes:

- a. Crie um StorageClass com o parâmetro Trident.NetApp.io/replicação: 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 MirrorRelationship CR 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"

```

O Provisioner criará um relacionamento SnapMirror com o nome da política de relacionamento configurado (ou padrão para ONTAP) e inicializará-o.

- c. Crie um PVC com StorageClass criado anteriormente para atuar como secundário (destino SnapMirror).

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
```

O Astra Control Provisioner verificará o CRD de relacionamento do tridentMirrorRelationship e falhará em criar o volume se o relacionamento não existir. Se o relacionamento existir, o Supervisor de Controle Astra garantirá que o novo FlexVol volume seja colocado em um SVM que seja emparelhado com o SVM remoto definido no espelhamento.

Estados de replicação de volume

Um relacionamento de espelhamento do Trident (TMR) é um CRD que representa um fim de uma relação de replicação entre PVCs. O TMR de destino tem um estado, que diz ao Astra Control Provisioner qual é o estado desejado. O TMR de destino tem os seguintes estados:

- *** Estabelecido***: O PVC local é o volume de destino de uma relação de espelho, e esta é uma nova relação.
- **Promovido**: O PVC local é ReadWrite e montável, sem relação de espelho atualmente em vigor.
- *** Restabelecido***: O PVC local é o volume de destino de uma relação de espelho e também estava anteriormente nessa relação de espelho.
 - O estado restabelecido deve ser usado se o volume de destino estiver em uma relação com o volume de origem, porque ele sobrescreve o conteúdo do volume de destino.
 - O estado restabelecido falhará se o volume não estiver previamente em uma relação com a fonte.

Promover PVC secundário durante um failover não planejado

Execute a seguinte etapa no cluster secundário do Kubernetes:

- Atualize o campo `spec.State` do TrigentMirrorRelationship para `promoted`.

Promover PVC secundário durante um failover planejado

Durante um failover planejado (migração), execute as seguintes etapas para promover o PVC secundário:

Passos

1. No cluster primário do Kubernetes, crie um snapshot do PVC e aguarde até que o snapshot seja criado.
2. No cluster principal do Kubernetes, 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 secundário do Kubernetes, atualize o campo *spec.State* do *tridentMirrorRelationship* CR para *promoted* e *spec.promotedSnapshotHandle* para ser o *internalName* do snapshot.
4. No cluster secundário do Kubernetes, confirme o status (campo *status.State*) do *TridentMirrorRelationship* para promovido.

Restaurar uma relação de espelhamento após um failover

Antes de restaurar uma relação de espelho, escolha o lado que você deseja fazer como o novo primário.

Passos

1. No cluster secundário do Kubernetes, certifique-se de que os valores do campo *spec.remoteVolumeHandle* no *TridentMirrorRelationship* sejam atualizados.
2. No cluster secundário do Kubernetes, atualize o campo *spec.mirror* do *TridentMirrorRelationship* para *reestablished*.

Operações adicionais

O Astra Control Provisioner dá suporte às seguintes operações nos volumes primário e secundário:

Replique PVC primário para um novo PVC secundário

Certifique-se de que você já tem um PVC primário e um PVC secundário.

Passos

1. Exclua as CRDs *PersistentVolumeClaim* e *TridentMirrorRelationship* do cluster secundário (destino) estabelecido.
2. Exclua o CRD do *tridentMirrorRelationship* do cluster primário (de origem).
3. Crie um novo CRD de *TridentMirrorRelationship* no cluster primário (de origem) para o novo PVC secundário (de destino) que você deseja estabelecer.

Redimensione um PVC espelhado, primário ou secundário

O PVC pode ser redimensionado como normal, o ONTAP irá expandir automaticamente qualquer destino flexvols se a quantidade de dados exceder o tamanho atual.

Remova a replicação de um PVC

Para remover a replicação, execute uma das seguintes operações no volume secundário atual:

- Exclua o MirrorRelationship no PVC secundário. Isso quebra a relação de replicação.
- Ou atualize o campo spec.State para *promovido*.

Excluir um PVC (que foi anteriormente espelhado)

O Astra Control Provisioner verifica se há PVCs replicados e libera a relação de replicação antes de tentar excluir o volume.

Eliminar um TMR

A exclusão de um TMR em um lado de um relacionamento espelhado faz com que o TMR restante passe para o estado *promovido* antes que o Astra Control Provisioner conclua a exclusão. Se o TMR selecionado para exclusão já estiver no estado *promovido*, não há relacionamento de espelhamento existente e o TMR será removido e o Astra Control Provisioner promoverá o PVC local para *ReadWrite*. Essa exclusão libera os metadados do SnapMirror para o volume local no ONTAP. Se este volume for usado em uma relação de espelho no futuro, ele deve usar um novo TMR com um estado de replicação de volume *established* ao criar a nova relação de espelho.

Atualizar relações de espelho quando o ONTAP estiver online

As relações de espelho podem ser atualizadas a qualquer momento depois que são estabelecidas. Pode utilizar os `state: promoted` campos ou `state: reestablished` 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 espelho quando o ONTAP estiver offline

Você pode usar um CRD para executar uma atualização do SnapMirror sem que o Astra Control tenha conectividade direta com o cluster do ONTAP. Consulte o seguinte formato de exemplo do TrigentActionMirrorUpdate:

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 CRD do TrigentActionMirrorUpdate. Ele pode tomar um valor de *successful*, *in progress* ou *Failed*.

Habilite o Astra Control Provisioner

As versões 23,10 e posteriores do Trident incluem a opção de usar o Astra Control Provisioner, que permite que usuários licenciados do Astra Control acessem o recurso avançado de provisionamento de storage. O Astra Control Provisioner fornece essa funcionalidade estendida, além da funcionalidade padrão baseada em CSI Astra Trident. Você pode usar este procedimento para ativar e instalar o Astra Control Provisioner.

Sua assinatura do Astra Control Service inclui automaticamente a licença para uso do Astra Control Provisioner.

Nas próximas atualizações do Astra Control, o parceiro Astra Control substituirá o Astra Trident como provisionador de storage e orquestrador e será obrigatório para uso do Astra Control. Por causa disso, é altamente recomendável que os usuários do Astra Control ativem o Astra Control Provisioner. O Astra Trident continuará a ser de código aberto e será lançado, mantido, suportado e atualizado com o novo CSI e outros recursos do NetApp.

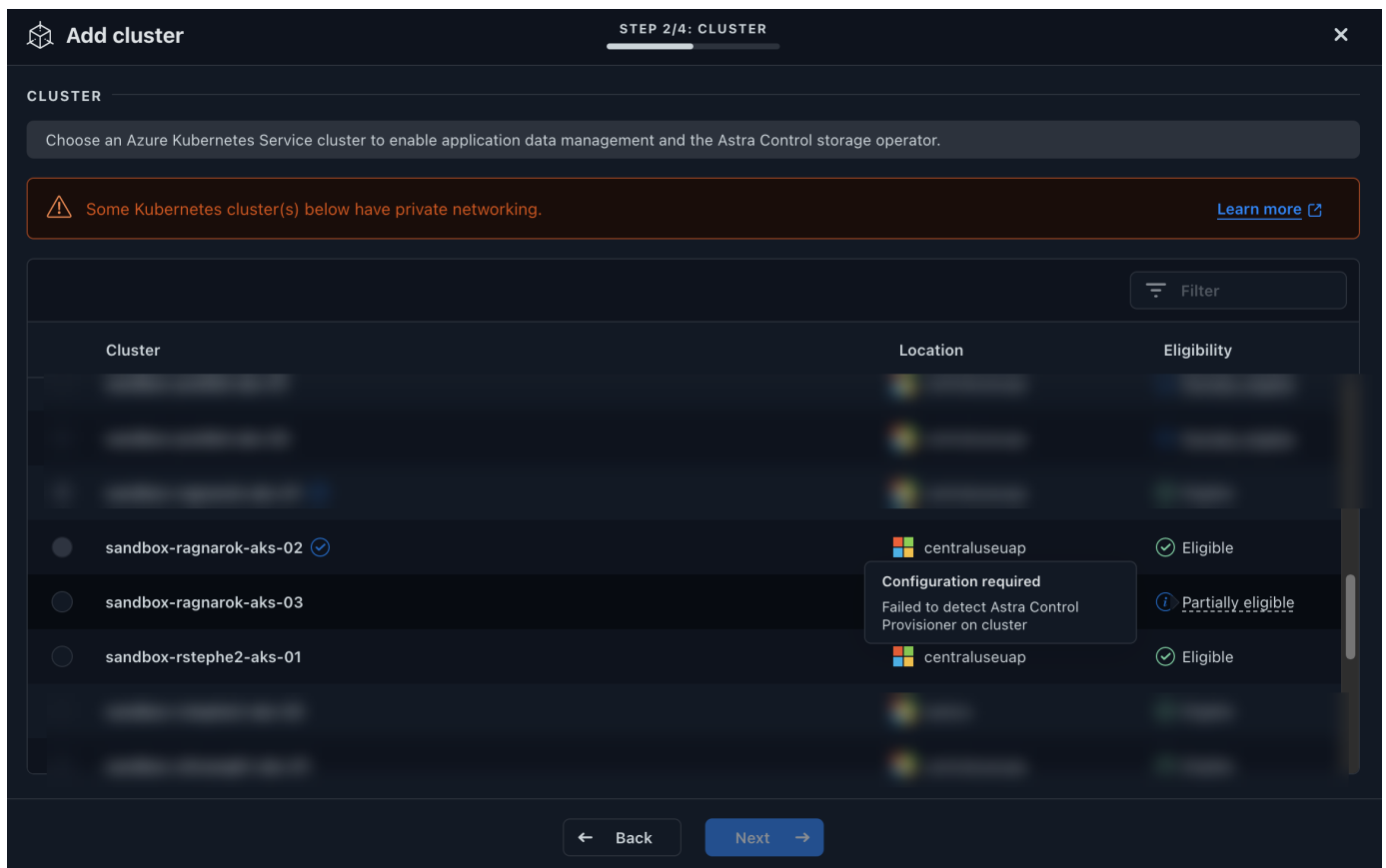
Como sei se preciso habilitar o Astra Control Provisioner?

Se você adicionar um cluster ao Astra Control Service que não tenha o Astra Trident instalado anteriormente, o cluster será marcado como `Eligible`. Depois de ["Adicione o cluster ao Astra Control"](#) você , o Astra Control Provisioner será ativado automaticamente.

Se o cluster não estiver marcado `Eligible`, ele será marcado `Partially eligible` por uma das seguintes opções:

- Ele está usando uma versão mais antiga do Astra Trident
- Ele está usando um Astra Trident 23,10 que ainda não tem a opção de provisionador habilitada
- É um tipo de cluster que não permite a ativação automática

Para `Partially eligible` casos, use estas instruções para ativar manualmente o Astra Control Provisioner para seu cluster.



Antes de ativar o Astra Control Provisioner

Se você já tiver um Astra Trident sem o parceiro Astra Control e quiser habilitar o parceiro Astra Control, faça o seguinte primeiro:

- **Se você tiver o Astra Trident instalado, confirme que sua versão está dentro de uma janela de quatro versões:** Você pode fazer uma atualização direta para o Astra Trident 24,02 com o Astra Control Provisioner se o seu Astra Trident estiver dentro de uma janela de quatro versões da versão 24,02. Por exemplo, você pode fazer o upgrade diretamente do Astra Trident 23,04 para o 24,02.
- **Confirme que seu cluster tem uma arquitetura de sistema AMD64:** A imagem Astra Control Provisioner é fornecida em arquiteturas de CPU AMD64 e ARM64, mas apenas AMD64 é compatível com Astra Control.

Passos

1. Acesse o Registro de imagem do NetApp Astra Control:
 - a. Faça login na IU do Astra Control Service e Registre sua ID de conta do Astra Control.
 - i. Selecione o ícone de figura no canto superior direito da página.
 - ii. Selecione **Acesso à API**.
 - iii. Anote o seu ID de conta.
 - b. Na mesma página, selecione **Generate API token** e copie a cadeia de token da API para a área de transferência e salve-a no seu editor.
 - c. Faça login no Registro Astra Control usando seu método preferido:

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

```
crane auth login cr.astra.netapp.io -u <account-id> -p <api-token>
```

2. (Apenas registros personalizados) siga estes passos para mover a imagem para o seu registro personalizado. Se você não estiver usando um Registro, siga as etapas do operador Trident na [próxima seção](#).



Você pode usar Podman em vez de Docker para os seguintes comandos. Se você estiver usando um ambiente Windows, o PowerShell é recomendado.

Docker

- a. Extraia a imagem Astra Control Provisioner do Registro:



A imagem puxada não suportará múltiplas plataformas e só suportará a mesma plataforma que o host que puxou a imagem, como o Linux AMD64.

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0  
--platform <cluster platform>
```

Exemplo:

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0  
--platform linux/amd64
```

- b. Marque a imagem:

```
docker tag cr.astra.netapp.io/astra/trident-acp:24.02.0  
<my_custom_registry>/trident-acp:24.02.0
```

- c. Envie a imagem para o seu registro personalizado:

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

Grua

- a. Copie o manifesto Astra Control Provisioner para o seu Registro personalizado:

```
crane copy cr.astra.netapp.io/astra/trident-acp:24.02.0  
<my_custom_registry>/trident-acp:24.02.0
```

3. Determine se o método de instalação original do Astra Trident usou um.

4. Ative o Astra Control Provisioner no Astra Trident usando o método de instalação que você usou originalmente:

Operador do Astra Trident

- a. "Baixe o instalador do Astra Trident e extraia-o".
- b. Siga estas etapas se você ainda não tiver instalado o Astra Trident ou se tiver removido o operador da sua implantação original do Astra Trident:
 - i. Crie o CRD:

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.1
6.yaml
```

- ii. Crie o namespace Trident (`kubectl create namespace trident`) ou confirme se o namespace Trident ainda existe (`kubectl get all -n trident`). Se o namespace tiver sido removido, crie-o novamente.
- c. Atualize o Astra Trident para 24.06.0:



Para clusters que executam o Kubernetes 1,24 ou anterior, `bundle_pre_1_25.yaml` use o . Para clusters que executam o Kubernetes 1,25 ou posterior, `bundle_post_1_25.yaml` use o .

```
kubectl -n trident apply -f trident-installer/deploy/<bundle-
name.yaml>
```

- d. Verifique se o Astra Trident está em execução:

```
kubectl get torc -n trident
```

Resposta:

NAME	AGE
trident	21m

- e. se você tem um Registro que usa segredos, crie um segredo para usar para puxar a imagem Astra Control Provisioner:

```
kubectl create secret docker-registry <secret_name> -n trident
--docker-server=<my_custom_registry> --docker-username=<username>
--docker-password=<token>
```

- f. Edite o TridentOrchestrator CR e faça as seguintes edições:

```
kubectl edit torc trident -n trident
```

- i. Defina um local de Registro personalizado para a imagem Astra Trident ou extraia-a do Registro Astra Control (`tridentImage: <my_custom_registry>/trident:24.02.0`ou `tridentImage: netapp/trident:24.06.0`).
- ii. Ative o Astra Control Provisioner (`enableACP: true`).
- iii. Defina o local de Registro personalizado para a imagem Astra Control Provisioner ou extraia-a do Registro Astra Control (`acpImage: <my_custom_registry>/trident-acp:24.02.0`ou `acpImage: cr.astra.netapp.io/astra/trident-acp:24.02.0`).
- iv. Se tiver estabelecido [a imagem puxa segredos](#) anteriormente neste procedimento, pode defini-los aqui (`imagePullSecrets: - <secret_name>`). Use o mesmo nome secreto que você estabeleceu nas etapas anteriores.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  tridentImage: <registry>/trident:24.06.0
  enableACP: true
  acpImage: <registry>/trident-acp:24.06.0
  imagePullSecrets:
    - <secret_name>
```

- g. Salve e saia do arquivo. O processo de implantação começará automaticamente.
- h. Verifique se o operador, a implantação e as replicaset são criados.

```
kubectl get all -n trident
```



Deve haver apenas **uma instância** do operador em um cluster do Kubernetes. Não crie várias implantações do operador Astra Trident.

- i. Verifique se o `trident-acp` contentor está em execução e se `acpVersion` está 24.02.0 com um status de Installed:

```
kubectl get torc -o yaml
```

Resposta:

```
status:
  acpVersion: 24.02.0
  currentInstallationParams:
    ...
    acpImage: <registry>/trident-acp:24.02.0
    enableACP: "true"
    ...
  ...
status: Installed
```

tridentctl

- "Baixe o instalador do Astra Trident e extraia-o".
- "Se você tiver um Astra Trident existente, desinstale-o do cluster que o hospeda".
- Instalar o Astra Trident com a previsão de controle Astra ativada (`--enable-acp=true`):

```
./tridentctl -n trident install --enable-acp=true --acp
-image=mycustomregistry/trident-acp:24.02
```

- Confirme se o Astra Control Provisioner foi ativado:

```
./tridentctl -n trident version
```

Resposta:

```
+-----+-----+-----+ | SERVER
VERSION | CLIENT VERSION | ACP VERSION | +-----+
+-----+-----+-----+ | 24.02.0 | 24.02.0 | 24.02.0. |
+-----+-----+-----+
```

Leme

- Se tiver o Astra Trident 23.07.1 ou anterior instalado, "**desinstalar**" o operador e outros componentes.
- Se o cluster do Kubernetes estiver executando o 1,24 ou anterior, exclua a psp:

```
kubectl delete psp tridentoperatorpod
```

- Adicione o repositório Astra Trident Helm:


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

d. Atualize o gráfico Helm:

```
helm repo update netapp-trident
```

Resposta:

```
Hang tight while we grab the latest from your chart
repositories...
...Successfully got an update from the "netapp-trident" chart
repository
Update Complete. ☐Happy Helming!☐
```

e. Liste as imagens:

```
./tridentctl images -n trident
```

Resposta:

```
| v1.28.0           | netapp/trident:24.06.0|
|                   | docker.io/netapp/trident-
autosupport:24.06|
|                   | registry.k8s.io/sig-storage/csi-
provisioner:v4.0.0|
|                   | registry.k8s.io/sig-storage/csi-
attacher:v4.5.0|
|                   | registry.k8s.io/sig-storage/csi-
resizer:v1.9.3|
|                   | registry.k8s.io/sig-storage/csi-
snapshotter:v6.3.3|
|                   | registry.k8s.io/sig-storage/csi-node-
driver-registrar:v2.10.0 |
|                   | netapp/trident-operator:24.06.0 (optional)
```

f. Certifique-se de que o Trident-Operator 24.06.0 está disponível:

```
helm search repo netapp-trident/trident-operator --versions
```

Resposta:

NAME	CHART VERSION	APP VERSION	
DESCRIPTION			
netapp-trident/trident-operator	100.2406.0	24.06.0	A

g. Utilize `helm install` e execute uma das seguintes opções que incluem estas definições:

- Um nome para o local de implantação
- A versão Astra Trident
- O nome da imagem Astra Control Provisioner
- A bandeira para habilitar o provisionador
- (Opcional) Um caminho de Registro local. Se você estiver usando um Registro local, o ["Imagens de Trident"](#) pode estar localizado em um Registro ou Registros diferentes, mas todas as imagens CSI devem estar localizadas no mesmo Registro.
- O namespace Trident

Opções

- Imagens sem registro

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=cr.astra.netapp.io/astra/trident-
acp:24.06.0 --set enableACP=true --set operatorImage=netapp/trident-
operator:24.06.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.06
--set tridentImage=netapp/trident:24.06.0 --namespace trident
```

- Imagens em um ou mais Registros

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=<your-registry>:<acp image> --set
enableACP=true --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=netapp/trident-operator:24.06.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.06
--set tridentImage=netapp/trident:24.06.0 --namespace trident
```

Você pode usar `helm list` para revisar detalhes de instalação, como nome, namespace, gráfico, status, versão do aplicativo e número de revisão.

Se você tiver algum problema na implantação do Trident usando o Helm, execute este comando para desinstalar completamente o Astra Trident:

```
./tridentctl uninstall -n trident
```

Não ["Remova completamente CRDS Astra Trident"](#) como parte da sua desinstalação antes de tentar ativar o Astra Control Provisioner novamente.

Resultado

A funcionalidade Astra Control Provisioner está ativada e você pode usar todos os recursos disponíveis para a versão em execução.

Depois que o Astra Control Provisioner for instalado, o cluster que hospeda o provisionador na IU do Astra Control mostrará um `ACP version` número de versão instalado em vez `Trident version` de campo e atual.

CLUSTER STATUS

Available

Version v1.24.9+rke2r2	Managed 2024/03/15 17:32 UTC	Kube-system namespace UID <div></div>	ACP Version <div></div>
Private route identifier <div>...</div>	Cloud instance private	Default bucket astra-bucket1 (inherited)	

Overview

Namespaces

Storage

Activity

Para mais informações

- ["O Astra Trident atualiza a documentação"](#)

Use a topologia CSI

O Astra Trident pode criar e anexar volumes de forma seletiva a nós presentes em um cluster Kubernetes usando o ["Recurso de topologia CSI"](#).

Visão geral

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

Saiba mais sobre o recurso de topologia de CSI ["aqui"](#) .

O Kubernetes oferece dois modos exclusivos de vinculação de volume:

- `VolumeBindingMode`Com o definido como `Immediate`, o Astra Trident cria o volume sem qualquer reconhecimento de topologia. A vinculação de volume e o provisionamento dinâmico são tratados quando o PVC é criado. Esse é o padrão `VolumeBindingMode` e é adequado para clusters que não impõem restrições de topologia. Os volumes persistentes são criados sem depender dos requisitos de agendamento do pod solicitante.

- Com `VolumeBindingMode` definido como `WaitForFirstConsumer`, a criação e a vinculação de um volume persistente para um PVC é adiada até que um pod que usa o PVC seja programado e criado. Dessa forma, os volumes são criados para atender às restrições de agendamento impostas pelos requisitos de topologia.



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

O que você vai precisar

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

- Um cluster de Kubernetes executando um ["Versão do Kubernetes compatível"](#)

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

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

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

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

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

Aqui está um exemplo de definição de backend:

YAML

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

JSON

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



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

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

```

---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: 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, as `region` etiquetas e `zone` representam a localização do conjunto de armazenamento. `topology.kubernetes.io/region` `topology.kubernetes.io/zone` e `workload` são as etiquetas de onde os pools de storage podem ser consumidos.

Etapas 2: Defina StorageClasses que estejam cientes da topologia

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

Veja o exemplo a seguir:

```

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

```

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

Passo 3: Criar e usar um PVC

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

Veja o exemplo spec abaixo:

```

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

```

Criar um PVC usando este manifesto resultaria no seguinte:


```

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

```

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

```

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

```

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

Veja a seguinte saída:

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

Atualize os backends para incluir `supportedTopologies`

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

Encontre mais informações

- ["Gerenciar recursos para contêineres"](#)
- ["NodeSelector"](#)
- ["Afinidade e anti-afinidade"](#)
- ["Taints e Tolerations"](#)

Trabalhar com instantâneos

Os snapshots de volume do Kubernetes de volumes persistentes (PVS) permitem cópias pontuais de volumes. Você pode criar um snapshot de um volume criado usando o Astra Trident, importar um snapshot criado fora do Astra Trident, criar um novo volume a partir de um snapshot existente e recuperar dados de volume de snapshots.

Visão geral

O instantâneo de volume é suportado por `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, e `azure-netapp-files` drivers.

Antes de começar

Você deve ter um controlador de snapshot externo e definições personalizadas de recursos (CRDs) para trabalhar com snapshots. Essa é a responsabilidade do orquestrador do Kubernetes (por exemplo: Kubeadm, GKE, OpenShift).

Se a distribuição do Kubernetes não incluir a controladora de snapshot e CRDs, [Implantar um controlador de snapshot de volume](#) consulte .



Não crie um controlador de snapshot se estiver criando instantâneos de volume sob demanda em um ambiente GKE. O GKE usa um controlador instantâneo oculto integrado.

Criar um instantâneo de volume

Passos

1. Criar um `VolumeSnapshotClass`. para obter mais informações, "[VolumeSnapshotClass](#)" consulte .
 - O driver aponta para o condutor Astra Trident CSI.
 - `deletionPolicy` pode ser `Delete` ou `Retain`. Quando definido como `Retain`, o instantâneo físico subjacente no cluster de armazenamento é retido mesmo quando o `VolumeSnapshot` objeto é excluído.

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 instantâneo de um PVC existente.

Exemplos

- Este exemplo cria um instantâneo 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 instantâneo de volume para um PVC chamado `pvc1` e o nome do instantâneo 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
```

- Pode identificar o `VolumeSnapshotContent` objeto para o `pvc1-snap` `VolumeSnapshot` descrevendo-o. O `Snapshot Content Name` identifica o objeto `VolumeSnapshotContent` que serve este instantâneo. O `Ready To Use` parâmetro indica que o instantâneo pode ser usado para criar um novo PVC.

```
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` nomeado `<pvc-name>` como a fonte dos dados. Depois que o PVC é criado, ele pode ser anexado a um pod e usado como qualquer outro PVC.



O PVC será criado no mesmo backend que o volume de origem. ["KB: A criação de um PVC a partir de um instantâneo de PVC do Trident não pode ser criada em um back-end alternativo"](#)Consulte a .

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

Importar um instantâneo de volume

O Astra Trident é compatível com o ["Processo de snapshot pré-provisionado do Kubernetes"](#) para permitir que o administrador do cluster crie um `VolumeSnapshotContent` objeto e importe snapshots criados fora do Astra Trident.

Antes de começar

O Astra Trident precisa ter criado ou importado o volume pai do snapshot.

Passos

1. **Cluster admin:** Crie um `VolumeSnapshotContent` objeto que faça referência ao snapshot de back-end. Isso inicia o fluxo de trabalho de snapshot no Astra Trident.
 - Especifique o nome do instantâneo de back-end em annotations as `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 Astra Trident pelo snapshotter externo na `ListSnapshots` chamada.



O `<volumeSnapshotContentName>` nem sempre pode corresponder ao nome do instantâneo do back-end devido a restrições de nomenclatura CR.

Exemplo

O exemplo a seguir cria um `VolumeSnapshotContent` objeto que faz referência a snapshot de back-end `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>

```

2. **Cluster admin:** Crie o VolumeSnapshot CR que faz referência ao VolumeSnapshotContent objeto. Isso solicita acesso para usar o VolumeSnapshot em um namespace dado.

Exemplo

O exemplo a seguir cria um VolumeSnapshot CR chamado import-snap que faz referência ao VolumeSnapshotContent import-snap-content chamado .

```

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. O Astra Trident cria o TridentSnapshot.
 - O snapshotter externo define VolumeSnapshotContent para readyToUse e VolumeSnapshot para true.
 - Trident retorna readyToUse=true.
4. **Qualquer usuário:** Crie um PersistentVolumeClaim para fazer referência ao novo VolumeSnapshot, onde o spec.dataSource nome (ou spec.dataSourceRef) é o VolumeSnapshot nome.

Exemplo

O exemplo a seguir cria um PVC referenciando o VolumeSnapshot nome import-snap.

```

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

```

Recuperar dados de volume usando snapshots

O diretório instantâneo é oculto por padrão para facilitar a compatibilidade máxima dos volumes provisionados usando os `ontap-nas drivers` e `ontap-nas-economy`. Ative o `.snapshot` diretório para recuperar dados de instantâneos diretamente.

Use a CLI do ONTAP de restauração de snapshot de volume para restaurar um volume para um estado gravado em um snapshot anterior.

```

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

```



Quando você restaura uma cópia snapshot, a configuração de volume existente é sobrescrita. As alterações feitas aos dados de volume após a criação da cópia instantânea são perdidas.

O diretório instantâneo é oculto por padrão para facilitar a compatibilidade máxima dos volumes provisionados usando os `ontap-nas drivers` e `ontap-nas-economy`. Ative o `.snapshot` diretório para recuperar dados de instantâneos diretamente.



Quando você restaura uma cópia snapshot, a configuração de volume existente é sobrescrita. As alterações feitas aos dados de volume após a criação da cópia instantânea são perdidas.

Restauração de volume no local a partir de um instantâneo

O Astra Control Provisioner fornece restauração rápida de volume no local a partir de um snapshot usando o `TridentActionSnapshotRestore (TASR)` CR. Esse CR funciona como uma ação imperativa do Kubernetes e não persiste após a conclusão da operação.

O Astra Control Provisioner oferece suporte à restauração de snapshot no `ontap-san` `ontap-san-economy` , , `ontap-nas` `ontap-nas-flexgroup` , , , , `azure-netapp-files` `gcp-cvs`, e `solidfire-san drivers`.

Antes de começar

Você deve ter um PVC vinculado e instantâneo de volume disponível.

- Verifique se o status do PVC está vinculado.

```
kubectl get pvc
```

- Verifique se o instantâneo do volume está pronto para ser usado.

```
kubectl get vs
```

Passos

1. Crie o TASR CR. Este exemplo cria um CR para instantâneo de PVC `pvc1` e volume `pvc1-snapshot`.

```
cat tasr-pvc1-snapshot.yaml

apiVersion: v1
kind: TridentActionSnapshotRestore
metadata:
  name: this-doesnt-matter
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. Aplique o CR para restaurar a partir do instantâneo. Este exemplo restaura do instantâneo `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml

tridentactionsnapshotrestore.trident.netapp.io/this-doesnt-matter
created
```

Resultados

O Astra Control Provisioner restaura os dados do snapshot. Você pode verificar o status de restauração de snapshot.

```
kubectl get tasr -o yaml

apiVersion: v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: this-doesnt-matter
    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: ""
```



- Na maioria dos casos, o Astra Control Provisioner não tentará automaticamente a operação em caso de falha. Terá de efetuar novamente a operação.
- Os usuários do Kubernetes sem acesso de administrador podem ter permissão para que o administrador crie um TASR CR em seu namespace de aplicativo.

Eliminar um PV com instantâneos associados

Ao excluir um volume persistente com snapshots associados, o volume Trident correspondente é atualizado para um "estado de exclusão". Remova os snapshots de volume para excluir o volume Astra Trident.

Implantar um controlador de snapshot de volume

Se a sua distribuição do Kubernetes não incluir a controladora de snapshot e CRDs, você poderá implantá-los da seguinte forma.

Passos

1. Criar CRDs de instantâneos de volume.

```
cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-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.yaml
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 instantâneo.

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



Se necessário, abra `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e atualize `namespace` para o seu namespace.

Links relacionados

- ["Instantâneos de volume"](#)
- ["VolumeSnapshotClass"](#)

Informações sobre direitos autorais

Copyright © 2026 NetApp, Inc. Todos os direitos reservados. Impresso nos EUA. Nenhuma parte deste documento protegida por direitos autorais pode ser reproduzida de qualquer forma ou por qualquer meio — gráfico, eletrônico ou mecânico, incluindo fotocópia, gravação, gravação em fita ou storage em um sistema de recuperação eletrônica — sem permissão prévia, por escrito, do proprietário dos direitos autorais.

O software derivado do material da NetApp protegido por direitos autorais está sujeito à seguinte licença e isenção de responsabilidade:

ESTE SOFTWARE É FORNECIDO PELA NETAPP "NO PRESENTE ESTADO" E SEM QUAISQUER GARANTIAS EXPRESSAS OU IMPLÍCITAS, INCLUINDO, SEM LIMITAÇÕES, GARANTIAS IMPLÍCITAS DE COMERCIALIZAÇÃO E ADEQUAÇÃO A UM DETERMINADO PROPÓSITO, CONFORME A ISENÇÃO DE RESPONSABILIDADE DESTES DOCUMENTOS. EM HIPÓTESE ALGUMA A NETAPP SERÁ RESPONSÁVEL POR QUALQUER DANO DIRETO, INDIRETO, INCIDENTAL, ESPECIAL, EXEMPLAR OU CONSEQUENCIAL (INCLUINDO, SEM LIMITAÇÕES, AQUISIÇÃO DE PRODUTOS OU SERVIÇOS SOBRESSAIENTES; PERDA DE USO, DADOS OU LUCROS; OU INTERRUPÇÃO DOS NEGÓCIOS), INDEPENDENTEMENTE DA CAUSA E DO PRINCÍPIO DE RESPONSABILIDADE, SEJA EM CONTRATO, POR RESPONSABILIDADE OBJETIVA OU PREJUÍZO (INCLUINDO NEGLIGÊNCIA OU DE OUTRO MODO), RESULTANTE DO USO DESTES DOCUMENTOS, MESMO SE ADVERTIDA DA RESPONSABILIDADE DE TAL DANO.

A NetApp reserva-se o direito de alterar quaisquer produtos descritos neste documento, a qualquer momento e sem aviso. A NetApp não assume nenhuma responsabilidade nem obrigação decorrentes do uso dos produtos descritos neste documento, exceto conforme expressamente acordado por escrito pela NetApp. O uso ou a compra deste produto não representam uma licença sob quaisquer direitos de patente, direitos de marca comercial ou quaisquer outros direitos de propriedade intelectual da NetApp.

O produto descrito neste manual pode estar protegido por uma ou mais patentes dos EUA, patentes estrangeiras ou pedidos pendentes.

LEGENDA DE DIREITOS LIMITADOS: o uso, a duplicação ou a divulgação pelo governo estão sujeitos a restrições conforme estabelecido no subparágrafo (b)(3) dos Direitos em Dados Técnicos - Itens Não Comerciais no DFARS 252.227-7013 (fevereiro de 2014) e no FAR 52.227- 19 (dezembro de 2007).

Os dados aqui contidos pertencem a um produto comercial e/ou serviço comercial (conforme definido no FAR 2.101) e são de propriedade da NetApp, Inc. Todos os dados técnicos e software de computador da NetApp fornecidos sob este Contrato são de natureza comercial e desenvolvidos exclusivamente com despesas privadas. O Governo dos EUA tem uma licença mundial limitada, irrevogável, não exclusiva, intransferível e não sublicenciável para usar os Dados que estão relacionados apenas com o suporte e para cumprir os contratos governamentais desse país que determinam o fornecimento de tais Dados. Salvo disposição em contrário no presente documento, não é permitido usar, divulgar, reproduzir, modificar, executar ou exibir os dados sem a aprovação prévia por escrito da NetApp, Inc. Os direitos de licença pertencentes ao governo dos Estados Unidos para o Departamento de Defesa estão limitados aos direitos identificados na cláusula 252.227-7015(b) (fevereiro de 2014) do DFARS.

Informações sobre marcas comerciais

NETAPP, o logotipo NETAPP e as marcas listadas em <http://www.netapp.com/TM> são marcas comerciais da NetApp, Inc. Outros nomes de produtos e empresas podem ser marcas comerciais de seus respectivos proprietários.