Documentação do Astra Trident 22,04

Astra Trident

NetApp
December 03, 2024

This PDF was generated from https://docs.netapp.com/pt-br/trident-2204/index.html on December 03, 2024. Always check docs.netapp.com for the latest.

Índice

Documentação do Astra Trident 22,04 . . . 1
Notas de versão . . . 2
- Novidades em 22,04 . . . 2
- Mudanças em 22.01.1 . . . 3
- Mudanças em 22.01.0 . . . 3
- Mudanças em 21.10.1 . . . 3
- Mudanças em 21.10.0 . . . 4
- Problemas conhecidos . . . 5
- Encontre mais informações . . . 6

Conceitos . . . 7
- Introdução ao Astra Trident . . . 7
- Controladores ONTAP . . . 8
- Provisionamento . . . 8
- Instantâneos de volume . . . 9
- Pools de storage virtuais . . . 10
- Grupos de acesso de volume . . . 11

Comece agora . . . 12
- Experimente . . . 12
- Requisitos . . . 12
- Visão geral da implantação . . . 16
- Implante com o operador Trident . . . 19
- Implante com o tridentctl . . . 28
- O que vem a seguir? . . . 31

Gerenciar o Astra Trident . . . 37
- Atualizar o Astra Trident . . . 37
- Atualize com o operador . . . 38
- Atualize com o tridentctl . . . 46
- Desinstale o Astra Trident . . . 49
- Downgrade Astra Trident . . . 51

Use o Astra Trident . . . 55
- Configurar backends . . . 55
- Crie backends com kubectl . . . 122
- Execute o gerenciamento de back-end com o kubectl . . . 129
- Execute o gerenciamento de back-end com o tridentctl . . . 130
- Alternar entre opções de gerenciamento de back-end . . . 132
- Gerenciar classes de armazenamento . . . 138
- Executar operações de volume . . . 140
- Prepare o nó de trabalho . . . 166
- Preparação automática do nó de trabalho . . . 170
- Monitore o Astra Trident . . . 170

Astra Trident para Docker . . . 175
- Pré-requisitos para implantação . . . 175
- Implante o Astra Trident . . . 178

- Atualize ou desinstale o Astra Trident . . . 182
- Trabalhe com volumes . . . 184
- Recolher registos . . . 193
- Gerenciar várias instâncias do Astra Trident . . . 194
- Opções de configuração de armazenamento . . . 195
- Problemas e limitações conhecidos . . . 210

Perguntas frequentes . . . 212
- Questões gerais . . . 212
- Instalar e usar o Astra Trident em um cluster Kubernetes . . . 212
- Solução de problemas e suporte . . . 214
- Atualizar o Astra Trident . . . 215
- Gerenciar backends e volumes . . . 215

Suporte . . . 221

Solução de problemas . . . 222
- Resolução de problemas gerais . . . 222
- Solução de problemas de uma implantação de Trident mal sucedida usando o operador . . . 224
- Solução de problemas de uma implantação do Trident mal sucedida usando `tridentctl` . . . 226

Práticas recomendadas e recomendações . . . 227
- Implantação . . . 227
- Configuração de armazenamento . . . 227
- Integre o Astra Trident . . . 234
- Proteção de dados . . . 245
- Segurança . . . 250

Referência . . . 252
- Portas Astra Trident . . . 252
- API REST do Astra Trident . . . 252
- Opções de linha de comando . . . 253
- Produtos da NetApp integrados ao Kubernetes . . . 254
- Objetos Kubernetes e Trident . . . 255
- comandos e opções tridentctl . . . 267

Versões anteriores da documentação . . . 273

Avisos legais . . . 274
- Direitos de autor . . . 274
- Marcas comerciais . . . 274
- Patentes . . . 274
- Política de privacidade . . . 274
- Código aberto . . . 274

Documentação do Astra Trident 22,04

Notas de versão

As Notas de versão fornecem informações sobre novos recursos, aprimoramentos e correções de bugs na versão mais recente do Astra Trident.

> O `tridentctl` binário para Linux que é fornecido no arquivo zip do instalador é a versão testada e suportada. Esteja ciente de que o `macos` binário fornecido na `/extras` parte do arquivo zip não é testado ou suportado.

Novidades em 22,04

A NetApp está continuamente melhorando e aprimorando seus produtos e serviços. Aqui estão alguns dos recursos mais recentes do Astra Trident. Para versões anteriores, "Versões anteriores da documentação"consulte .

> Se você estiver atualizando de qualquer versão anterior do Trident e usar o Azure NetApp Files, o `location` parâmetro config agora é um campo único obrigatório.

Correções

- Análise melhorada de nomes de iniciadores iSCSI. (Edição [n.º 681](https://github.com/NetApp/trident/issues/681))
- Corrigido problema em que os parâmetros da classe de armazenamento CSI não eram permitidos. (Edição [n.º 598](https://github.com/NetApp/trident/issues/598))
- Declaração de chave duplicada corrigida no CRD Trident. (Edição [n.º 671](https://github.com/NetApp/trident/issues/671))
- Corrigidos registos de instantâneos do CSI imprecisos. (Edição [n.º 629](https://github.com/NetApp/trident/issues/629))
- Corrigido o problema com a remoção de volumes em nós excluídos. (Edição [n.º 691](https://github.com/NetApp/trident/issues/691))
- Adição de manipulação de inconsistências de sistema de arquivos em dispositivos de bloco. (Edição [n.º 656](https://github.com/NetApp/trident/issues/656))
- Corrigido problema ao puxar imagens de suporte automático ao definir o `imageRegistry` sinalizador durante a instalação. (Edição [n.º 715](https://github.com/NetApp/trident/issues/715))
- Corrigido o problema em que o driver do ANF não conseguiu clonar um volume com várias regras de exportação.

Melhorias

- As conexões de entrada para os endpoints seguros da Trident agora exigem um mínimo de TLS 1,3. (Edição [n.º 698](https://github.com/NetApp/trident/issues/698))
- O Trident agora adiciona cabeçalhos HSTS às respostas de seus endpoints seguros.
- O Trident agora tenta ativar o recurso de permissões unix do Azure NetApp Files automaticamente.
- **Kubernetes**: O daemonset do Trident agora é executado na classe de prioridade crítica do nó do sistema. (Edição [n.º 694](https://github.com/NetApp/trident/issues/694))

Remoções

O driver da série e (desativado desde 20,07) foi removido.

Mudanças em 22.01.1

Correções

- Corrigido o problema com a remoção de volumes em nós excluídos. ("GitHub Edição nº 691")
- Corrigido o pânico ao acessar campos nil para espaço agregado nas respostas da API do ONTAP.

Mudanças em 22.01.0

Correções

- **Kubernetes:** aumente o tempo de repetição do backoff do Registro de nós para clusters grandes.
- Corrigido problema em que o driver azure-NetApp-Files poderia ser confundido por vários recursos com o mesmo nome.
- Os LIFs de dados SAN IPv6 da ONTAP agora funcionam se especificados com colchetes.
- Corrigido o problema em que a tentativa de importar um volume já importado retorna EOF deixando PVC em estado pendente. ("GitHub Edição nº 489")
- Corrigido o problema quando a performance do Astra Trident diminui quando são criados snapshots > 32 em um volume SolidFire.
- Substituído SHA-1 por SHA-256 na criação de certificado SSL.
- Driver do ANF fixo para permitir nomes duplicados de recursos e limitar operações a um único local.
- Driver do ANF fixo para permitir nomes duplicados de recursos e limitar operações a um único local.

Melhorias

- Melhorias do Kubernetes:
  - Adicionado suporte para Kubernetes 1,23.
  - Adicione opções de agendamento para pods Trident quando instalado via Operador Trident ou Helm. ("GitHub Edição nº 651")
- Permitir volumes entre regiões no driver do GCP. ("GitHub Edição nº 633")
- Adicionado suporte para a opção 'unixPermissions' aos volumes do ANF. ("GitHub Edição nº 666")

Desvalorizações

A interface REST do Trident pode ouvir e servir apenas em endereços 127.0.0.1 ou [::1]

Mudanças em 21.10.1

A versão v21.10.0 tem um problema que pode colocar o controlador Trident em um estado CrashLoopBackOff quando um nó é removido e depois adicionado de volta ao cluster do Kubernetes. Esse problema foi corrigido no v21,10.1 (GitHub Issue 669).

Correções

- Condição de corrida potencial fixa ao importar um volume em um back-end CVS do GCP, resultando em falha na importação.
- Corrigido um problema que pode colocar o controlador Trident em um estado CrashLoopBackOff quando um nó é removido e depois adicionado de volta ao cluster do Kubernetes (problema 669 do GitHub).
- Corrigido o problema em que os SVMs não eram mais descobertos se nenhum nome SVM foi especificado (problema 612 do GitHub).

Mudanças em 21.10.0

Correções

- Corrigido o problema em que clones de volumes XFS não podiam ser montados no mesmo nó que o volume de origem (problema 514 do GitHub).
- Corrigido o problema em que o Astra Trident registrou um erro fatal no desligamento (problema 597 do GitHub).
- Correções relacionadas ao Kubernetes:
  - Retorne o espaço usado de um volume como o mínimo restoresSize ao criar snapshots com `ontap-nas` drivers e `ontap-nas-flexgroup` (GitHub Issue 645).
  - Corrigido o problema em que `Failed to expand filesystem` o erro foi registrado após o redimensionamento de volume (GitHub problema 560).
  - Corrigido o problema em que um pod poderia ficar preso `Terminating` no estado (GitHub problema 572).
  - Corrigido o caso em que um `ontap-san-economy` FlexVol pode estar cheio de LUNs instantâneos (GitHub problema 533).
  - Corrigido o problema do instalador personalizado YAML com imagem diferente (problema 613 do GitHub).
  - Corrigido cálculo do tamanho do instantâneo (GitHub edição 611).
  - Corrigido o problema em que todos os instaladores do Astra Trident podiam identificar o Kubernetes simples como OpenShift (problema 639 do GitHub).
  - Corrigido o operador do Trident para parar a reconciliação se o servidor da API do Kubernetes não estiver acessível (problema 599 do GitHub).

Melhorias

- Adicionado suporte à `unixPermissions` opção para volumes de performance do GCP-CVS.
- Adicionado suporte para volumes CVS otimizados para escala no GCP na faixa de 600 GiB a 1 TIB.
- Aprimoramentos relacionados ao Kubernetes:
  - Adicionado suporte para Kubernetes 1,22.
  - Habilitou o operador do Trident e o gráfico Helm para trabalhar com o Kubernetes 1,22 (GitHub Issue 628).
  - Adicionado a imagem do operador ao `tridentctl` comando imagens (GitHub Issue 570).

Melhorias experimentais

- Adicionado suporte para replicação de volume no `ontap-san` driver.
- Adicionado suporte REST **Tech Preview** para os `ontap-nas-flexgroup` drivers , `ontap-san`, e `ontap-nas-economy` .

Problemas conhecidos

Problemas conhecidos identificam problemas que podem impedi-lo de usar o produto com sucesso.

- Agora, o Astra Trident aplica um espaço em `fsType` (`fsType=""`branco) para volumes que não têm o `fsType` especificado em seu StorageClass. Ao trabalhar com o Kubernetes 1,17 ou posterior, a Trident dá suporte a fornecer um espaço em branco `fsType` para volumes NFS. Para volumes iSCSI, é necessário definir o `fsType` no StorageClass ao aplicar um `fsGroup` contexto de uso de segurança.
- Ao usar um back-end em várias instâncias do Astra Trident, cada arquivo de configuração de back-end deve ter um valor diferente `storagePrefix` para backends do ONTAP ou usar um diferente `TenantName` para backends do SolidFire. O Astra Trident não consegue detectar volumes que outras instâncias do Astra Trident criaram. Tentar criar um volume existente em backends ONTAP ou SolidFire é bem-sucedido, porque o Astra Trident trata a criação de volume como uma operação idempotente. Se `storagePrefix` ou `TenantName` não forem diferentes, pode haver colisões de nomes para volumes criados no mesmo back-end.
- Ao instalar o Astra Trident (usando `tridentctl` ou o Operador Trident) e usar `tridentctl` para gerenciar o Astra Trident, você deve garantir que a `KUBECONFIG` variável de ambiente esteja definida. Isso é necessário para indicar o cluster do Kubernetes com `tridentctl` quem trabalhar. Ao trabalhar com vários ambientes do Kubernetes, você deve garantir que o `KUBECONFIG` arquivo seja obtido com precisão.
- Para executar a recuperação de espaço on-line para PVS iSCSI, o SO subjacente no nó de trabalho pode exigir que as opções de montagem sejam passadas para o volume. Isso é verdade para instâncias RHEL/RedHat CoreOS, que exigem o `discard` "opção de montagem"; Certifique-se de que a opção Descartar mountOption está incluída no seu[`StorageClass`site para suportar descarte de blocos online.
- Se você tiver mais de uma instância do Astra Trident por cluster Kubernetes, o Astra Trident não poderá se comunicar com outras instâncias e não poderá descobrir outros volumes que eles criaram, o que leva a um comportamento inesperado e incorreto se mais de uma instância for executada em um cluster. Só deve haver uma instância do Astra Trident por cluster Kubernetes.
- Se os objetos baseados no Astra Trident `StorageClass` forem excluídos do Kubernetes enquanto o Astra Trident estiver off-line, o Astra Trident não removerá as classes de storage correspondentes de seu banco de dados quando ele voltar on-line. Você deve excluir essas classes de armazenamento usando `tridentctl` ou a API REST.
- Se um usuário excluir um PV provisionado pelo Astra Trident antes de excluir o PVC correspondente, o Astra Trident não excluirá automaticamente o volume de backup. Você deve remover o volume via `tridentctl` ou a API REST.
- A ONTAP não pode provisionar simultaneamente mais de um FlexGroup de cada vez, a menos que o conjunto de agregados seja exclusivo para cada solicitação de provisionamento.
- Ao usar o Astra Trident mais de IPv6 TB, você deve especificar `managementLIF` e `dataLIF` na definição de back-end entre colchetes. Por exemplo, `[fd20:8b1e:b258:2000:f816:3eff:feec:0]`.
- Se estiver usando `solidfire-san` o driver com OpenShift 4,5, certifique-se de que os nós de trabalho subjacentes usem MD5 como o algoritmo de autenticação CHAP.

Encontre mais informações

- "Astra Trident no GitHub"
- "Blogs do Astra Trident"

Conceitos

Introdução ao Astra Trident

O Astra Trident é um projeto de código aberto totalmente compatível mantido pela NetApp como parte "Família de produtos Astra"do . Ele foi desenvolvido para ajudar você a atender às demandas de persistência das aplicações em contêineres usando interfaces padrão do setor, como a Container Storage Interface (CSI).

O Astra Trident implanta em clusters Kubernetes como pods e fornece serviços de orquestração de storage dinâmico para seus workloads do Kubernetes. Ele permite que suas aplicações em contêiner consumam de forma rápida e fácil o armazenamento persistente do amplo portfólio do NetApp, que inclui o ONTAP (AFF/FAS/Select/Cloud/Amazon FSX for NetApp ONTAP), o software Element (NetApp HCI/SolidFire), o armazenamento de dados Astra e o serviço Azure NetApp Files e o Cloud Volumes Service no Google Cloud.

O Astra Trident também é uma tecnologia básica para o Astra do NetApp, que atende aos casos de uso de proteção de dados, recuperação de desastres, portabilidade e migração para workloads Kubernetes. Com a tecnologia de gerenciamento de dados líder do setor da NetApp para snapshots, backups, replicação e clonagem.

Arquiteturas de cluster Kubernetes compatíveis

O Astra Trident é compatível com as seguintes arquiteturas de Kubernetes:

Arquiteturas de cluster do Kubernetes	Suportado	Instalação predefinida
Único mestre, computação	Sim	Sim
Mestre múltiplo, computação	Sim	Sim
Mestre `etcd`, , computação	Sim	Sim
Mestre, infraestrutura, computação	Sim	Sim

O que é o Astra?

Com o Astra, é mais fácil para as empresas gerenciar, proteger e mover workloads em contêineres com muitos dados executados no Kubernetes dentro e entre nuvens públicas e no local. O Astra provisiona e fornece storage de contêiner persistente usando o Astra Trident do portfólio de storage comprovado e expansivo da NetApp, na nuvem pública e no local. Ele também oferece um conjunto avançado de recursos avançados de gerenciamento de dados com reconhecimento de aplicações, como snapshot, backup e restauração, logs de atividade e clonagem ativa para proteção de dados, recuperação de desastres/dados, auditoria de dados e casos de uso de migração para workloads Kubernetes.

Você pode se inscrever para uma avaliação gratuita na página Astra.

Para mais informações

- "Família de produtos NetApp Astra"
- "Documentação do Astra Control Service"

- "Documentação do Astra Control Center"
- "Documentação do Astra Data Store"
- "Documentação da API Astra"

Controladores ONTAP

O Astra Trident fornece cinco drivers de storage ONTAP exclusivos para comunicação com clusters ONTAP. Saiba mais sobre como cada driver lida com a criação de volumes e controle de acesso e suas capacidades.

Condutor	Protocolo	Modo de volume	Modos de acesso suportados	Sistemas de arquivos suportados
`ontap-nas`	NFS	Sistema de ficheiros	RWO, RWX, ROX	"", nfs
`ontap-nas-economy`	NFS	Sistema de ficheiros	RWO, RWX, ROX	"", nfs
`ontap-nas-flexgroup`	NFS	Sistema de ficheiros	RWO, RWX, ROX	"", nfs
`ontap-san`	ISCSI	Bloco	RWO, ROX, RWX	Sem sistema de ficheiros. Dispositivo de bloco bruto
`ontap-san`	ISCSI	Sistema de ficheiros	RWO, ROX	`xfs` `ext3`, , `ext4`
`ontap-san-economy`	ISCSI	Bloco	RWO, ROX, RWX	Sem sistema de ficheiros. Dispositivo de bloco bruto
`ontap-san-economy`	ISCSI	Sistema de ficheiros	RWO, ROX	`xfs` `ext3`, , `ext4`

Os backends do ONTAP podem ser autenticados usando credenciais de login para uma função de segurança (nome de usuário/senha) ou usando a chave privada e o certificado que está instalado no cluster do ONTAP. Você pode atualizar os backends existentes para passar de um modo de autenticação para o outro com `tridentctl update backend`o .

Provisionamento

O provisionamento no Astra Trident tem duas fases primárias. A primeira fase associa uma classe de armazenamento ao conjunto de conjuntos de armazenamento de back-end adequados e ocorre como uma preparação necessária antes do provisionamento. A segunda fase inclui a própria criação de volume e requer a escolha de um pool de armazenamento daqueles associados à classe de armazenamento do volume pendente.

A associação de pools de storage de back-end a uma classe de armazenamento depende dos atributos solicitados da classe de armazenamento e `storagePools` das listas , `additionalStoragePools` e `excludeStoragePools` . Quando você cria uma classe de storage, o Trident compara os atributos e pools

oferecidos por cada um de seus back-ends aos solicitados pela classe de storage. Se os atributos e o nome de um pool de storage corresponderem a todos os atributos e nomes de pool solicitados, o Astra Trident adicionará esse pool de storage ao conjunto de pools de storage adequados para essa classe de storage. Além disso, o Astra Trident adiciona todos os pools de storage listados na `additionalStoragePools` lista a esse conjunto, mesmo que seus atributos não atendam a todos ou a qualquer um dos atributos solicitados da classe de storage. Você deve usar a `excludeStoragePools` lista para substituir e remover pools de armazenamento de uso para uma classe de armazenamento. O Astra Trident executa um processo semelhante sempre que você adiciona um novo back-end, verificando se os pools de storage atendem às classes de storage existentes e removendo quaisquer que tenham sido marcados como excluídos.

Em seguida, o Astra Trident usa as associações entre classes de storage e pools de storage para determinar onde provisionar volumes. Quando você cria um volume, o Astra Trident primeiro obtém o conjunto de pools de storage para a classe de storage desse volume e, se você especificar um protocolo para o volume, o Astra Trident removerá esses pools de storage que não podem fornecer o protocolo solicitado (por exemplo, um back-end NetApp HCI/SolidFire não poderá fornecer um volume baseado em arquivo enquanto um back-end do ONTAP nas não puder fornecer um volume baseado em bloco). O Astra Trident aleatoriza a ordem desse conjunto resultante, para facilitar uma distribuição uniforme de volumes e, em seguida, iterá-lo, tentando provisionar o volume em cada pool de storage por sua vez. Se for bem-sucedido em um, ele retorna com sucesso, registrando quaisquer falhas encontradas no processo. O Astra Trident retorna uma falha **somente se** falhar ao provisionamento em **todos** dos pools de storage disponíveis para a classe de storage e o protocolo solicitados.

Instantâneos de volume

Saiba mais sobre como o Astra Trident lida com a criação de snapshots de volume para seus drivers.

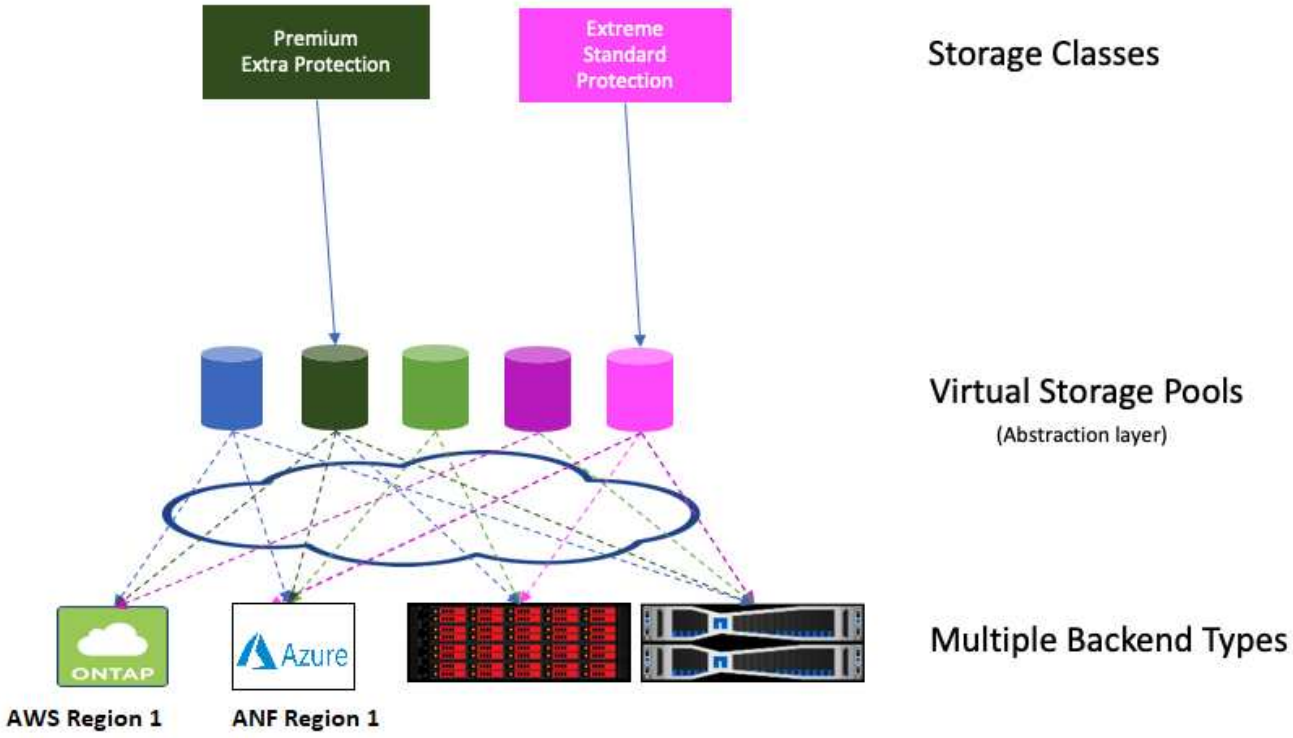
- Para os `ontap-nas` drivers , `ontap-san`, , `gcp-cvs` e , `azure-netapp-files` cada volume persistente (PV) é mapeado para um FlexVol. Como resultado, os snapshots de volume são criados como snapshots do NetApp. A tecnologia Snapshot da NetApp oferece mais estabilidade, escalabilidade, capacidade de recuperação e desempenho do que as tecnologias de snapshot da concorrência. Essas cópias snapshot são extremamente eficientes no tempo necessário para criá-las e no espaço de storage.
- Para `ontap-nas-flexgroup` o condutor, cada volume persistente (PV) é mapeado para um FlexGroup. Como resultado, os snapshots de volume são criados como snapshots do NetApp FlexGroup. A tecnologia Snapshot da NetApp oferece mais estabilidade, escalabilidade, capacidade de recuperação e desempenho do que as tecnologias de snapshot da concorrência. Essas cópias snapshot são extremamente eficientes no tempo necessário para criá-las e no espaço de storage.
- Para `ontap-san-economy` o driver, os PVS mapeiam para LUNs criados em FlexVols compartilhados. VolumeSnapshots de PVS são obtidos executando FlexClones do LUN associado. A tecnologia FlexClone da ONTAP possibilita a criação de cópias até dos maiores conjuntos de dados quase instantaneamente. As cópias compartilham blocos de dados com os pais, não consumindo storage, exceto o necessário para os metadados.
- Para `solidfire-san` o driver, cada PV mapeia para um LUN criado no cluster do software/NetApp HCI do NetApp Element. VolumeSnapshots são representados por instantâneos de elementos do LUN subjacente. Esses snapshots são cópias pontuais e ocupam apenas um pequeno espaço e recursos do sistema.
- Ao trabalhar com `ontap-nas` os drivers e `ontap-san`, os snapshots do ONTAP são cópias pontuais do FlexVol e consomem espaço no próprio FlexVol. Isso pode resultar na quantidade de espaço gravável no volume para reduzir com o tempo, à medida que os snapshots são criados/programados. Uma maneira simples de lidar com isso é aumentar o volume redimensionando pelo Kubernetes. Outra opção é excluir snapshots que não são mais necessários. Quando um VolumeSnapshot criado por meio do Kubernetes é excluído, o Astra Trident excluirá o snapshot associado do ONTAP. Os snapshots do ONTAP que não foram criados pelo Kubernetes também podem ser excluídos.

Com o Astra Trident, você pode usar o VolumeSnapshots para criar novos PVS a partir deles. A criação de PVS a partir desses snapshots é realizada usando a tecnologia FlexClone para backends ONTAP e CVS compatíveis. Ao criar um PV a partir de um instantâneo, o volume de backup é um FlexClone do volume pai do instantâneo. O `solidfire-san` driver usa clones de volume do software Element para criar PVS a partir de snapshots. Aqui ele cria um clone a partir do snapshot do elemento.

Pools de storage virtuais

Os pools de storage virtuais fornecem uma camada de abstração entre os back-ends de storage do Astra Trident e o Kubernetes' `StorageClasses`. Eles permitem que um administrador defina aspetos, como localização, desempenho e proteção para cada back-end de uma maneira comum e independente de back-end, sem `StorageClass` especificar qual backend físico, pool de back-end ou tipo de back-end usar para atender aos critérios desejados.

O administrador de storage pode definir pools de armazenamento virtual em qualquer um dos backends do Astra Trident em um arquivo de definição JSON ou YAML.



Qualquer aspeto especificado fora da lista de pools virtuais é global para o back-end e se aplicará a todos os pools virtuais, enquanto cada pool virtual pode especificar um ou mais aspetos individualmente (substituindo quaisquer aspetos globais de back-end).

> Ao definir pools de armazenamento virtual, não tente reorganizar a ordem dos pools virtuais existentes em uma definição de back-end. Também é aconselhável não editar/modificar atributos para um pool virtual existente e definir um novo pool virtual.

A maioria dos aspetos são especificados em termos específicos de back-end. Fundamentalmente, os valores de aspeto não são expostos fora do driver do back-end e não estão disponíveis para correspondência em

`StorageClasses`. em vez disso, o administrador define um ou mais rótulos para cada pool virtual. Cada rótulo é um par chave:valor, e os rótulos podem ser comuns em backends exclusivos. Assim como aspetos, os rótulos podem ser especificados por pool ou globais para o back-end. Ao contrário de aspetos, que têm nomes e valores predefinidos, o administrador tem total discrição para definir chaves de rótulo e valores conforme necessário.

A `StorageClass` identifica qual pool virtual usar fazendo referência aos rótulos dentro de um parâmetro seletor. Os seletores de pool virtual suportam os seguintes operadores:

Operador	Exemplo	O valor do rótulo de um pool deve:
`=`	desempenho superior	Correspondência
`!=`	performance! extrema	Não corresponde
`in`	localização em (leste, oeste)	Esteja no conjunto de valores
`notin`	notificação de desempenho (prata, bronze)	Não estar no conjunto de valores
`<key>`	proteção	Existe com qualquer valor
`!<key>`	!proteção	Não existe

Grupos de acesso de volume

Saiba mais sobre como o Astra Trident usa "grupos de acesso de volume".

> Ignore esta seção se você estiver usando CHAP, que é recomendado para simplificar o gerenciamento e evitar o limite de escala descrito abaixo. Além disso, se você estiver usando Astra Trident no modo CSI, você pode ignorar esta seção. O Astra Trident usa o CHAP quando instalado como um provisionador aprimorado de CSI.

O Astra Trident pode usar grupos de acesso a volumes para controlar o acesso aos volumes provisionados. Se o CHAP estiver desativado, ele espera encontrar um grupo de acesso chamado `trident`, a menos que você especifique um ou mais IDs de grupo de acesso na configuração.

Embora o Astra Trident associe novos volumes ao(s) grupo(s) de acesso configurado(s), ele não cria nem gerencia grupos de acesso. Os grupos de acesso devem existir antes que o back-end de storage seja adicionado ao Astra Trident e precisam conter as IQNs de iSCSI de cada nó no cluster do Kubernetes que potencialmente poderia montar os volumes provisionados por esse back-end. Na maioria das instalações, isso inclui cada nó de trabalho no cluster.

Para clusters de Kubernetes com mais de 64 nós, você deve usar vários grupos de acesso. Cada grupo de acesso pode conter até 64 IQNs e cada volume pode pertencer a quatro grupos de acesso. Com o máximo de quatro grupos de acesso configurados, qualquer nó em um cluster de até 256 nós de tamanho poderá acessar qualquer volume. Para obter os limites mais recentes dos grupos de acesso de volume, "aqui" consulte .

Se você estiver modificando a configuração de uma que esteja usando o grupo de acesso padrão `trident` para outra que também use outras, inclua a ID do `trident` grupo de acesso na lista.

Comece agora

Experimente

O NetApp fornece uma imagem de laboratório pronta para uso que pode ser solicitada através "Unidade de teste do NetApp"do . O Test Drive oferece um ambiente sandbox que inclui um cluster de Kubernetes de três nós e o Astra Trident instalado e configurado. É uma ótima maneira de se familiarizar com o Astra Trident e explorar seus recursos.

Outra opção é ver o "Guia de Instalação do kubeadm" fornecido pelo Kubernetes.

> Você não deve usar o cluster do Kubernetes criado usando essas instruções em produção. Use os guias de implantação de produção fornecidos pela distribuição para criar clusters prontos para produção.

Se esta for a primeira vez que você estiver usando o Kubernetes, familiarize-se com os conceitos e as ferramentas "aqui".

Requisitos

Comece revisando os front-ends, backends e configuração de host suportados.

> Para saber mais sobre as portas que o Astra Trident usa, "aqui"consulte .

Frontens suportados (orquestradores)

O Astra Trident é compatível com vários mecanismos de contêiner e orquestradores, incluindo os seguintes:

- Anthos On-Prem (VMware) e Anthos em bare metal 1,9, 1,10, 1,11
- Kubernetes 1,18 - 1,24
- Mecanismo do Kubernetes do Mirantis 3,4
- OpenShift 4,7, 4,8, 4,9, 4,10

O operador Trident é suportado com estas versões:

- Anthos On-Prem (VMware) e Anthos em bare metal 1,9, 1,10, 1,11
- Kubernetes 1,18 - 1,24
- OpenShift 4,7, 4,8, 4,9, 4,10

O Astra Trident também trabalha com uma série de outras ofertas do Kubernetes totalmente gerenciadas e autogeridas, incluindo o Google Kubernetes Engine (GKE), o Amazon Elastic Kubernetes Services (EKS), o Azure Kubernetes Service (AKS), o Rancher e o portfólio VMware Tanzu.

Backends suportados (armazenamento)

Para usar o Astra Trident, você precisa de um ou mais dos seguintes back-ends compatíveis:

- Amazon FSX para NetApp ONTAP

- Azure NetApp Files
- Armazenamento de dados Astra
- Cloud Volumes ONTAP
- Cloud Volumes Service para GCP
- FAS/AFF/Selecione 9,3 ou posterior
- NetApp All SAN Array (ASA)
- Software NetApp HCI/Element 11 ou posterior

Requisitos de recursos

A tabela abaixo resume os recursos disponíveis com esta versão do Astra Trident e as versões do Kubernetes compatíveis.

Recurso	Versão do Kubernetes	É necessário ter portões?
CSI Trident	1,18 - 1,24	Não
Instantâneos de volume	1,18 - 1,24	Não
PVC a partir de instantâneos de volume	1,18 - 1,24	Não
Redimensionamento iSCSI PV	1,18 - 1,24	Não
ONTAP bidirectional CHAP	1,18 - 1,24	Não
Políticas de exportação dinâmica	1,18 - 1,24	Não
Operador Trident	1,18 - 1,24	Não
Preparação do nó de trabalho automático (beta)	1,18 - 1,24	Não
Topologia de CSI	1,18 - 1,24	Não

Sistemas operacionais de host testados

Por padrão, o Astra Trident é executado em um contentor e, portanto, será executado em qualquer trabalhador Linux. No entanto, esses funcionários precisam ser capazes de montar os volumes que o Astra Trident fornece usando o cliente NFS padrão ou iniciador iSCSI, dependendo dos backends que você está usando.

Embora o Astra Trident não "ofereça suporte" oficialmente a sistemas operacionais específicos, as seguintes distribuições Linux são conhecidas por funcionar:

- Versões do RedHat CoreOS (RHCOS) suportadas pela OpenShift Container Platform
- RHEL ou CentOS 7,4 ou posterior
- Ubuntu 18,04 ou posterior

O `tridentctl` utilitário também é executado em qualquer uma dessas distribuições do Linux.

Configuração de host

Dependendo do(s) back-end(s) em uso, os utilitários NFS e/ou iSCSI devem ser instalados em todos os trabalhadores do cluster. Consulte "aqui" para obter mais informações.

Configuração do sistema de storage

O Astra Trident pode exigir algumas alterações em um sistema de storage antes que uma configuração de back-end o use. "aqui"Consulte para obter detalhes.

Imagens de contêineres e versões correspondentes do Kubernetes

Para instalações com conexão de ar, a lista a seguir é uma referência das imagens de contêiner necessárias para instalar o Astra Trident. Use o `tridentctl images` comando para verificar a lista de imagens de contentor necessárias.

Versão do Kubernetes	Imagem do recipiente
v1.18.0	• NetApp/Trident:22.04.0 • NetApp/Trident-AutoSupport:22,04 • k8s.gcr.io/sig-storage/csi-provisionador:v2,2.2 • k8s.gcr.io/sig-storage/csi-attacher:v3,4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.4.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3,0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.5.0 • NetApp/Trident-operador: 22.04.0 (opcional)
v1.19.0	• NetApp/Trident:22.04.0 • NetApp/Trident-AutoSupport:22,04 • k8s.gcr.io/sig-storage/csi-provisionador:v2,2.2 • k8s.gcr.io/sig-storage/csi-attacher:v3,4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.4.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3,0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.5.0 • NetApp/Trident-operador: 22.04.0 (opcional)

Versão do Kubernetes	Imagem do recipiente
v1.20.0	• NetApp/Trident:22.04.0 • NetApp/Trident-AutoSupport:22,04 • k8s.gcr.io/sig-storage/csi-provisionador:v3,1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3,4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.4.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v5,0.1 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.5.0 • NetApp/Trident-operador: 22.04.0 (opcional)
v1.21.0	• NetApp/Trident:22.04.0 • NetApp/Trident-AutoSupport:22,04 • k8s.gcr.io/sig-storage/csi-provisionador:v3,1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3,4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.4.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v5,0.1 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.5.0 • NetApp/Trident-operador: 22.04.0 (opcional)
v1.22.0	• NetApp/Trident:22.04.0 • NetApp/Trident-AutoSupport:22,04 • k8s.gcr.io/sig-storage/csi-provisionador:v3,1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3,4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.4.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v5,0.1 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.5.0 • NetApp/Trident-operador: 22.04.0 (opcional)

Versão do Kubernetes	Imagem do recipiente
v1.23.0	• NetApp/Trident:22.04.0 • NetApp/Trident-AutoSupport:22,04 • k8s.gcr.io/sig-storage/csi-provisionador:v3,1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3,4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.4.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v5,0.1 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.5.0 • NetApp/Trident-operador: 22.04.0 (opcional)

No Kubernetes versão 1,20 e posterior, use a imagem validada `k8s.gcr.io/sig-storage/csi-snapshotter:v5.x` somente se a `v1` versão estiver servindo o `volumesnapshots.snapshot.storage.k8s.io` CRD. Se a `v1beta1` versão estiver servindo o CRD com/sem a `v1` versão, use a imagem validada `k8s.gcr.io/sig-storage/csi-snapshotter:v3.x`.

Visão geral da implantação

Você pode implantar o Astra Trident usando o operador Trident ou com `tridentctl`o .

A partir da versão 22,04, as chaves AES não serão mais regeneradas sempre que o Astra Trident for instalado. Com este lançamento, o Astra Trident instalará um novo objeto secreto que persiste em todas as instalações. Isso significa que `tridentctl` no 22,04 pode desinstalar versões anteriores do Trident, mas versões anteriores não podem desinstalar instalações do 22,04.

Escolha o método de implantação

Para determinar qual método de implantação usar, considere o seguinte:

Por que devo usar o operador Trident?

O "Operador Trident" é uma ótima maneira de gerenciar dinamicamente os recursos do Astra Trident e automatizar a fase de configuração. Existem alguns pré-requisitos que devem ser satisfeitos. "os requisitos"Consulte .

O operador Trident fornece vários benefícios, conforme descrito abaixo.

Funcionalidade de autorrecuperação

Você pode monitorar uma instalação do Astra Trident e tomar medidas ativamente para resolver problemas, como quando a implantação é excluída ou se for modificada acidentalmente. Quando o operador é configurado como uma implantação, um `trident-operator-<generated-id>` pod é criado. Este pod associa um `TridentOrchestrator` CR a uma instalação do Astra Trident e garante sempre que existe apenas uma ativa `TridentOrchestrator`. Em outras palavras, o operador garante que haja apenas uma instância do Astra Trident no cluster e controla sua configuração, garantindo que a instalação seja

idempotente. Quando as alterações são feitas na instalação (como, por exemplo, a exclusão do daemonset de implantação ou nó), o operador as identifica e as corrige individualmente.

Atualizações fáceis para instalações existentes

Você pode facilmente atualizar uma implantação existente com o operador. Você só precisa editar o `TridentOrchestrator` CR para fazer atualizações em uma instalação. Por exemplo, considere um cenário em que você precisa habilitar o Astra Trident para gerar logs de depuração.

Para fazer isso, corrija o `TridentOrchestrator` para definir `spec.debug` como `true`:

```
kubectl patch torc <trident-orchestrator-name> -n trident --type=merge -p
'{"spec":{"debug":true}}'
```

Após `TridentOrchestrator` a atualização, o operador processa as atualizações e corrige a instalação existente. Isso pode acionar a criação de novos pods para modificar a instalação de acordo.

Manipula automaticamente as atualizações do Kubernetes

Quando a versão do Kubernetes do cluster é atualizada para uma versão compatível, a operadora atualiza uma instalação existente do Astra Trident automaticamente e a altera para garantir que ela atenda aos requisitos da versão do Kubernetes.

> ⓘ Se o cluster for atualizado para uma versão não suportada, o operador impede a instalação do Astra Trident. Se o Astra Trident já tiver sido instalado com a operadora, um aviso será exibido para indicar que o Astra Trident está instalado em uma versão Kubernetes não suportada.

Gerencie clusters do Kubernetes usando o Cloud Manager

Com "Astra Trident usando o Cloud Manager"o , você pode atualizar para a versão mais recente do Astra Trident, adicionar e gerenciar classes de storage, conectá-las a ambientes de trabalho e fazer backup de volumes persistentes usando o Cloud Backup Service. O Cloud Manager oferece suporte à implantação do Astra Trident usando o operador Trident, manualmente ou usando o Helm.

Por que devo usar o Helm?

Se você tiver outras aplicações que está gerenciando usando o Helm, a partir do Astra Trident 21,01, poderá gerenciar sua implantação também usando o Helm.

Quando devo usar `tridenctl`?

Se você tiver uma implantação existente que deve ser atualizada ou se você estiver procurando personalizar altamente sua implantação, consulte "tridentctl"o uso do . Esse é o método convencional de implantação do Astra Trident.

Considerações para mover entre métodos de implantação

Não é difícil imaginar um cenário em que se deseja mover entre métodos de implantação. Você deve considerar o seguinte antes de tentar mover de uma `tridentctl` implantação para uma implantação baseada em operador ou vice-versa:

- Use sempre o mesmo método para desinstalar o Astra Trident. Se você implantou com `tridentctl`o ,

use a versão apropriada `tridentctl do binário para desinstalar o Astra Trident. Da mesma forma, se você estiver implantando com o operador, edite o `TridentOrchestrator` CR e configure `spec.uninstall=true` para desinstalar o Astra Trident.

- Se você tiver uma implantação baseada em operador que deseja remover e usar `tridentctl` para implantar o Astra Trident, primeiro edite `TridentOrchestrator` e configure `spec.uninstall=true` para desinstalar o Astra Trident. Em seguida, exclua `TridentOrchestrator` e a implantação do operador. Você pode instalar usando `tridentctl`o .
- Se você tiver uma implantação manual baseada em operador e quiser usar a implantação de operador Trident baseada em Helm, desinstale manualmente o operador primeiro e, em seguida, faça a instalação do Helm. Isso permite que o Helm implante o operador Trident com as etiquetas e anotações necessárias. Se você não fizer isso, sua implantação de operador Trident baseada em Helm falhará com erro de validação de rótulo e erro de validação de anotação. Se você tem uma `tridentctl` implantação baseada em -, você pode usar a implantação baseada em Helm sem problemas.

Entenda os modos de implantação

Há três maneiras de implantar o Astra Trident.

Implantação padrão

A implantação do Trident em um cluster Kubernetes resulta no instalador do Astra Trident fazendo duas coisas:

- A obter as imagens de contentor através da Internet
- Criação de um daemonset de implantação e/ou nó, que ativa pods do Astra Trident em todos os nós qualificados no cluster do Kubernetes.

Uma implantação padrão como essa pode ser realizada de duas maneiras diferentes:

- Utilização `tridentctl install`
- Utilizando o operador Trident. Você pode implantar o operador Trident manualmente ou usando o Helm.

Esse modo de instalação é a maneira mais fácil de instalar o Astra Trident e funciona para a maioria dos ambientes que não impõem restrições de rede.

Implantação off-line

Para executar uma implantação do AIR-gapped, você pode usar o `--image-registry` sinalizador ao chamar `tridentctl install` para apontar para um Registro de imagem privado. Se estiver implantando com o operador Trident, você poderá especificar `spec.imageRegistry` no `TridentOrchestrator`. Esse Registro deve conter as "Imagem Trident" imagens sidecar , "Imagem Trident AutoSupport" e CSI, conforme exigido pela versão do Kubernetes.

Para personalizar sua implantação, você pode usar `tridentctl` para gerar os manifestos para os recursos do Trident. Isso inclui a implantação, o daemonset, a conta de serviço e a função de cluster que o Astra Trident cria como parte de sua instalação.

Consulte esses links para obter mais informações sobre como personalizar sua implantação:

- "Personalize sua implantação baseada em operador"
*

Se você estiver usando um repositório de imagens privado, você deve adicionar `/sig-storage` ao final da URL do Registro privado. Ao usar um Registro privado para `tridentctl` implantação, você deve usar `--trident-image` e `--autosupport-image` em conjunto `--image-registry`com o . Se você estiver implantando o Astra Trident usando o operador Trident, verifique se o orquestrador CR inclui `tridentImage` e `autosupportImage` nos parâmetros de instalação.

Implantação remota

Aqui está uma visão geral de alto nível do processo de implantação remota:

- Implante a versão apropriada do `kubectl` na máquina remota de onde você deseja implantar o Astra Trident.
- Copie os arquivos de configuração do cluster do Kubernetes e defina a `KUBECONFIG` variável de ambiente na máquina remota.
- Inicie um `kubectl get nodes` comando para verificar se você pode se conetar ao cluster do Kubernetes necessário.
- Conclua a implementação a partir da máquina remota utilizando as etapas de instalação padrão.

Outras opções de configuração conhecidas

Ao instalar o Astra Trident em produtos do portfólio VMware Tanzu:

- O cluster precisa dar suporte a workloads privilegiados.
- A `--kubelet-dir` bandeira deve ser definida para a localização do diretório kubelet. Por padrão, isso é `/var/vcap/data/kubelet`.

  Especificar a localização do kubelet usando `--kubelet-dir` é conhecido por funcionar para o Operador Trident, Helm e `tridentctl` implantações.

Implante com o operador Trident

Você pode implantar o Astra Trident usando o operador Trident. Você pode implantar o operador Trident de duas maneiras:

- Usando o Trident "Carta do leme": o Gráfico Helm implanta o operador Trident e instala o Trident em uma etapa.
- Manualmente: O Trident fornece um "bundle.yaml" arquivo que pode ser usado para instalar o operador e criar objetos associados.

Se você ainda não se familiarizou com o "conceitos básicos", agora é um ótimo momento para fazer isso.

O que você vai precisar

Para implantar o Astra Trident, os seguintes pré-requisitos devem ser atendidos:

- Você tem Privileges completo para um cluster Kubernetes compatível com Kubernetes executando o Kubernetes 1,18 a 1,24.

- Você tem acesso a um sistema de storage NetApp compatível.
- Você tem a capacidade de montar volumes de todos os nós de trabalho do Kubernetes.
- Você tem um host Linux com `kubectl` (ou `oc`, se estiver usando o OpenShift) instalado e configurado para gerenciar o cluster do Kubernetes que deseja usar.
- Você definiu a `KUBECONFIG` variável de ambiente para apontar para a configuração do cluster do Kubernetes.
- Você ativou o "Portas de recurso exigidas pelo Astra Trident".
- Se você estiver usando o Kubernetes com Docker Enterprise "Siga os passos para ativar o acesso CLI", .

Tem tudo isso? Ótimo! Vamos começar.

Implante o operador Trident e instale o Astra Trident usando o Helm

Execute as etapas listadas para implantar o operador Trident usando Helm.

O que você vai precisar

Além dos pré-requisitos listados acima, para implantar o operador Trident usando o Helm, você precisa do seguinte:

- Kubernetes 1,18 - 1,24
- Helm versão 3

Passos

1. Adicionar o repositório Helm do Trident:

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

2. Use o `helm install` comando e especifique um nome para sua implantação. Veja o exemplo a seguir:

```
helm install <release-name> netapp-trident/trident-operator --version
22.4.0 --namespace <trident-namespace>
```

> Se você ainda não criou um namespace para Trident, você pode adicionar o `--create -namespace` parâmetro ao `helm install` comando. Helm irá então criar automaticamente o namespace para você.

Há duas maneiras de passar dados de configuração durante a instalação:

- `--values` (Ou `-f`): Especifique um arquivo YAML com substituições. Isso pode ser especificado várias vezes e o arquivo mais à direita terá precedência.
- `--set`: Especificar substituições na linha de comando.

Por exemplo, para alterar o valor padrão `debug` do , execute o seguinte `--set` comando:

```
$ helm install <name> netapp-trident/trident-operator --version 22.4.0
--set tridentDebug=true
```

O `values.yaml` arquivo, que faz parte do gráfico Helm, fornece a lista de chaves e seus valores padrão.

`helm list` mostra detalhes sobre a instalação, como nome, namespace, gráfico, status, versão do aplicativo, número de revisão e assim por diante.

Implante o operador Trident manualmente

Execute as etapas listadas para implantar manualmente o operador Trident.

Etapa 1: Qualifique seu cluster Kubernetes

A primeira coisa que você precisa fazer é fazer login no host Linux e verificar se ele está gerenciando um *working*, "Cluster compatível com Kubernetes" que você tem o Privileges necessário para.

> Com OpenShift, use `oc` em vez de `kubectl` em todos os exemplos que se seguem, e faça login como **system:admin** primeiro executando `oc login -u system:admin` ou `oc login -u kube-admin`.

Para verificar sua versão do Kubernetes, execute o seguinte comando:

```
kubectl version
```

Para ver se você tem o administrador do cluster do Kubernetes Privileges, execute o seguinte comando:

```
kubectl auth can-i '*' '*' --all-namespaces
```

Para verificar se você pode iniciar um pod que usa uma imagem do Docker Hub e alcançar seu sistema de armazenamento pela rede de pod, execute o seguinte comando:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Passo 2: Baixe e configure o operador

> A partir de 21,01, o operador Trident tem o escopo do cluster. O uso do operador Trident para instalar o Trident requer a criação da `TridentOrchestrator` Definição de recursos personalizada (CRD) e a definição de outros recursos. Você deve executar estas etapas para configurar o operador antes de poder instalar o Astra Trident.

1. Baixe a versão mais recente da "Pacote de instalação do Trident" na seção *Downloads* e extraia-a.

```
wget
https://github.com/NetApp/trident/releases/download/v22.04.0/trident-
installer-22.04.0.tar.gz
tar -xf trident-installer-22.04.0.tar.gz
cd trident-installer
```

2. Use o manifesto CRD apropriado para criar o `TridentOrchestrator` CRD. Em seguida, crie um `TridentOrchestrator` recurso personalizado mais tarde para instanciar uma instalação pelo operador.

   Execute o seguinte comando:

   ```
   kubectl create -f
   deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
   ```

3. Após a criação do `TridentOrchestrator` CRD, crie os seguintes recursos necessários para a implantação do operador:

   - Um ServiceAccount para o operador
   - Um ClusterRole e ClusterRoleBinding para o ServiceAccount
   - Uma PodSecurityPolicy dedicada
   - O próprio operador

     O instalador do Trident contém manifestos para definir esses recursos. Por padrão, o operador é implantado no `trident` namespace. Se o `trident` namespace não existir, use o manifesto a seguir para criar um.

   ```
   $ kubectl apply -f deploy/namespace.yaml
   ```

4. Para implantar o operador em um namespace diferente do namespace padrão `trident`, você deve atualizar o `serviceaccount.yaml`, `clusterrolebinding.yaml` e `operator.yaml` manifesta e gera o `bundle.yaml`.

   Execute o seguinte comando para atualizar os manifestos YAML e gerar o `bundle.yaml` usando o `kustomization.yaml`:

   ```
   kubectl kustomize deploy/ > deploy/bundle.yaml
   ```

   Execute o seguinte comando para criar os recursos e implantar o operador:

   ```
   kubectl create -f deploy/bundle.yaml
   ```

5. Para verificar o status do operador depois de ter implantado, faça o seguinte:

```
$ kubectl get deployment -n <operator-namespace>
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m

$ kubectl get pods -n <operator-namespace>
NAME                                READY   STATUS              RESTARTS
AGE
trident-operator-54cb664d-lnjxh   1/1     Running             0
3m
```

A implantação do operador cria com êxito um pod em execução em um dos nós de trabalho no cluster.

Deve haver apenas **uma instância** do operador em um cluster do Kubernetes. Não crie várias implantações do operador Trident.

Passo 3: Crie `TridentOrchestrator` e instale o Trident

Agora você está pronto para instalar o Astra Trident usando o operador! Isso exigirá a criação `TridentOrchestrator`do . O instalador do Trident vem com exemplos de definições para criar `TridentOrchestrator`. Isso inicia uma instalação no `trident` namespace.

```
$ kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

$ kubectl describe torc trident
Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:21.04
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Enable Node Prep:          false
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:21.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v21.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

O operador Trident permite personalizar a maneira como o Astra Trident é instalado usando os atributos na `TridentOrchestrator` especificação. "Personalize a implantação do Trident"Consulte .

O Status do `TridentOrchestrator` indica se a instalação foi bem-sucedida e exibe a versão do Trident instalado.

Estado	Descrição
A instalar	O operador está instalando o Astra Trident usando este `TridentOrchestrator` CR.
Instalado	O Astra Trident foi instalado com sucesso.
Desinstalação	O operador está desinstalando o Astra Trident, `spec.uninstall=true` porque .
Desinstalado	O Astra Trident foi desinstalado.
Falha	O operador não pôde instalar, corrigir, atualizar ou desinstalar o Astra Trident; o operador tentará recuperar automaticamente deste estado. Se este estado persistir, será necessário resolver o problema.
A atualizar	O operador está atualizando uma instalação existente.
Erro	O `TridentOrchestrator` não é utilizado. Outro já existe.

Durante a instalação, o status das `TridentOrchestrator` alterações de `Installing` para `Installed`. Se observar o `Failed` estado e o operador não conseguir recuperar sozinho, deve verificar os registos do operador. Consulte "solução de problemas"a secção .

Você pode confirmar se a instalação do Astra Trident foi concluída dando uma olhada nos pods criados:

```
$ kubectl get pod -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-7d466bf5c7-v4cpw        5/5     Running   0          1m
trident-csi-mr6zc                   2/2     Running   0          1m
trident-csi-xrp7w                   2/2     Running   0          1m
trident-csi-zh2jt                   2/2     Running   0          1m
trident-operator-766f7b8658-ldzsv   1/1     Running   0          3m
```

Você também pode usar `tridentctl` para verificar a versão do Astra Trident instalada.

```
$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.04.0        | 21.04.0        |
+----------------+----------------+
```

Agora você pode ir em frente e criar um backend. "tarefas pós-implantação"Consulte .

Para solucionar problemas durante a implantação, consulte "solução de problemas" a seção.

Personalizar a implantação do operador Trident

O operador Trident permite personalizar a maneira como o Astra Trident é instalado usando os atributos na `TridentOrchestrator` especificação.

Consulte a tabela a seguir para obter a lista de atributos:

Parâmetro	Descrição	Padrão
`namespace`	Namespace para instalar Astra Trident em	"predefinição"
`debug`	Habilite a depuração para o Astra Trident	falso
`IPv6`	Instalar o Astra Trident em IPv6	falso
`k8sTimeout`	Tempo limite para operações do Kubernetes	30sec
`silenceAutosupport`	Não envie pacotes AutoSupport para o NetApp automaticamente	falso
`enableNodePrep`	Gerenciar dependências de nó de trabalho automaticamente (**BETA**)	falso
`autosupportImage`	A imagem do recipiente para a telemetria AutoSupport	"NetApp/Trident-AutoSupport:21.04.0"
`autosupportProxy`	O endereço/porta de um proxy para o envio de telemetria AutoSupport	"http://proxy.example.com:8888"
`uninstall`	Um sinalizador usado para desinstalar o Astra Trident	falso
`logFormat`	Formato de log Astra Trident a ser usado [text,json]	"texto"
`tridentImage`	Imagem Astra Trident a instalar	"NetApp/Trident:21,04"
`imageRegistry`	Caminho para o Registro interno, do formato `<registry FQDN>[:port][/subpath]`	"k8s.gcr.io/sig-storage (mais de k8s 1,18 gb) ou quay.io/k8scsi gb"
`kubeletDir`	Caminho para o diretório kubelet no host	"/var/lib/kubelet"
`wipeout`	Uma lista de recursos a serem excluídos para realizar uma remoção completa do Astra Trident	
`imagePullSecrets`	Segredos para extrair imagens de um Registro interno	

Parâmetro	Descrição	Padrão
`controllerPluginNodeSelector`	Seletores de nós adicionais para pods executando o plug-in CSI controlador Trident. Segue o mesmo formato que pod.spec.nodeSelector.	Sem padrão; opcional
`controllerPluginTolerations`	Substitui as tolerâncias para pods que executam o plug-in CSI controlador Trident. Segue o mesmo formato que pod.spec.Tolerations.	Sem padrão; opcional
`nodePluginNodeSelector`	Seletores de nós adicionais para pods executando o plug-in CSI nó Trident. Segue o mesmo formato que pod.spec.nodeSelector.	Sem padrão; opcional
`nodePluginTolerations`	Substitui as tolerâncias para pods que executam o plug-in CSI nó Trident. Segue o mesmo formato que pod.spec.Tolerations.	Sem padrão; opcional

`spec.namespace` É especificado em `TridentOrchestrator` para indicar em que namespace Astra Trident está instalado. Este parâmetro **não pode ser atualizado depois que o Astra Trident é instalado**. Tentar fazê-lo faz com que o estado de `TridentOrchestrator` mude para `Failed`. O Astra Trident não deve ser migrado entre namespaces.

A preparação automática de nó de trabalho é um recurso **beta** destinado a ser usado apenas em ambientes não produtivos.

Para obter mais informações sobre a formatação dos parâmetros do pod, "Atribuindo pods a nós"consulte .

Você pode usar os atributos mencionados acima ao definir `TridentOrchestrator` para personalizar sua instalação. Aqui está um exemplo:

```
$ cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
```

Aqui está outro exemplo que mostra como o Trident pode ser implantado com seletores de nós:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  controllerPluginNodeSelector:
    nodetype: master
  nodePluginNodeSelector:
    storage: netapp
```

Se você estiver procurando personalizar a instalação além do que `TridentOrchestrator` os argumentos permitem, considere usar `tridentctl` para gerar manifestos YAML personalizados que você pode modificar conforme necessário.

Implante com o tridentctl

É possível implantar o Astra Trident usando `tridentctl`o .

Se você ainda não se familiarizou com o "conceitos básicos", agora é um ótimo momento para fazer isso.

Para personalizar sua implantação, "aqui"consulte .

O que você vai precisar

Para implantar o Astra Trident, os seguintes pré-requisitos devem ser atendidos:

- Você tem Privileges completo para um cluster compatível com Kubernetes.
- Você tem acesso a um sistema de storage NetApp compatível.
- Você tem a capacidade de montar volumes de todos os nós de trabalho do Kubernetes.
- Você tem um host Linux com `kubectl` (ou `oc`, se estiver usando o OpenShift) instalado e configurado para gerenciar o cluster do Kubernetes que deseja usar.
- Você definiu a `KUBECONFIG` variável de ambiente para apontar para a configuração do cluster do Kubernetes.
- Você ativou o "Portas de recurso exigidas pelo Astra Trident".
- Se você estiver usando o Kubernetes com Docker Enterprise "Siga os passos para ativar o acesso CLI", .

Tem tudo isso? Ótimo! Vamos começar.

Para obter informações sobre como personalizar sua implantação, "aqui"consulte .

Etapa 1: Qualifique seu cluster Kubernetes

A primeira coisa que você precisa fazer é fazer login no host Linux e verificar se ele está gerenciando um

working, "Cluster compatível com Kubernetes" que você tem o Privileges necessário para.

> Com o OpenShift, você usa `oc` em vez de `kubectl` em todos os exemplos que se seguem, e você deve fazer login como **system:admin** primeiro executando `oc login -u system:admin` ou `oc login -u kube-admin`.

Para verificar a versão do Kubernetes, execute o seguinte comando:

```
kubectl version
```

Para ver se você tem o administrador do cluster do Kubernetes Privileges, execute o seguinte comando:

```
kubectl auth can-i '*' '*' --all-namespaces
```

Para verificar se você pode iniciar um pod que usa uma imagem do Docker Hub e alcançar seu sistema de armazenamento pela rede de pod, execute o seguinte comando:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Identifique a versão do servidor Kubernetes. Você o usará quando instalar o Astra Trident.

Passo 2: Baixe e extraia o instalador

> O instalador do Trident cria um pod Trident, configura os objetos CRD que são usados para manter seu estado e inicializa os sidecars CSI que executam ações, como provisionar e anexar volumes aos hosts do cluster.

Você pode baixar a versão mais recente da "Pacote de instalação do Trident" na seção *Downloads* e extraí-la.

Por exemplo, se a versão mais recente for 21.07.1:

```
wget https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
tar -xf trident-installer-21.07.1.tar.gz
cd trident-installer
```

Etapa 3: Instale o Astra Trident

Instale o Astra Trident no namespace desejado executando o `tridentctl install` comando.

```
$ ./tridentctl install -n trident
....
INFO Starting Trident installation.                namespace=trident
INFO Created service account.
INFO Created cluster role.
INFO Created cluster role binding.
INFO Added finalizers to custom resource definitions.
INFO Created Trident service.
INFO Created Trident secret.
INFO Created Trident deployment.
INFO Created Trident daemonset.
INFO Waiting for Trident pod to start.
INFO Trident pod started.                          namespace=trident
pod=trident-csi-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=21.07.1
INFO Trident installation succeeded.
....
```

Será assim quando o instalador estiver completo. Dependendo do número de nós no cluster do Kubernetes, é possível observar mais pods:

```
$ kubectl get pod -n trident
NAME                            READY   STATUS    RESTARTS   AGE
trident-csi-679648bd45-cv2mx    4/4     Running   0          5m29s
trident-csi-vgc8n               2/2     Running   0          5m29s

$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.1        | 21.07.1        |
+----------------+----------------+
```

Se você vir saída semelhante ao exemplo acima, concluiu esta etapa, mas o Astra Trident ainda não está totalmente configurado. Vá em frente e continue para o próximo passo. "tarefas pós-implantação"Consulte .

No entanto, se o instalador não for concluído com êxito ou se você não vir um **Running** `trident-csi-<generated id>`, a plataforma não foi instalada.

Para solucionar problemas durante a implantação, consulte "solução de problemas" a seção.

Personalizar a implantação do tridentctl

O instalador do Trident permite personalizar atributos. Por exemplo, se você tiver copiado a imagem Trident

para um repositório privado, poderá especificar o nome da imagem `--trident-image` usando o . Se você copiou a imagem do Trident, bem como as imagens do sidecar do CSI necessárias para um repositório privado, pode ser preferível especificar a localização desse repositório usando o `--image-registry` switch, que assume o formulário `<registry FQDN>[:port]`.

Para que o Astra Trident configure automaticamente nós de trabalho para você, `--enable-node-prep` use o . Para obter mais detalhes sobre como funciona, "aqui"consulte .

> A preparação automática do nó de trabalho é um recurso **beta** destinado a ser usado apenas em ambientes não produtivos.

Se você estiver usando uma distribuição do Kubernetes, onde `kubelet` mantém seus dados em um caminho diferente do habitual `/var/lib/kubelet`, você poderá especificar o caminho alternativo usando `--kubelet-dir`o .

Se você precisar personalizar a instalação além do que os argumentos do instalador permitem, você também pode personalizar os arquivos de implantação. Usando o `--generate-custom-yaml` parâmetro cria os seguintes arquivos YAML no diretório do instalador `setup`:

- `trident-clusterrolebinding.yaml`
- `trident-deployment.yaml`
- `trident-crds.yaml`
- `trident-clusterrole.yaml`
- `trident-daemonset.yaml`
- `trident-service.yaml`
- `trident-namespace.yaml`
- `trident-serviceaccount.yaml`

Depois de gerar esses arquivos, você pode modificá-los de acordo com suas necessidades e, em seguida, usá-los `--use-custom-yaml` para instalar sua implantação personalizada.

```
./tridentctl install -n trident --use-custom-yaml
```

O que vem a seguir?

Depois de implantar o Astra Trident, você pode continuar criando um back-end, criando uma classe de storage, provisionando um volume e montando o volume em um pod.

Passo 1: Crie um backend

Agora você pode ir em frente e criar um back-end que será usado pelo Astra Trident para provisionar volumes. Para fazer isso, crie um `backend.json` arquivo que contenha os parâmetros necessários. Arquivos de configuração de exemplo para diferentes tipos de backend podem ser encontrados `sample-input` no diretório.

Consulte "aqui" para obter mais detalhes sobre como configurar o arquivo para o seu tipo de back-end.

```
cp sample-input/<backend template>.json backend.json
vi backend.json
```

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

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

```
./tridentctl -n trident logs
```

Depois de resolver o problema, basta voltar ao início deste passo e tentar novamente. Para obter mais dicas de solução de problemas, "o diagnóstico de avarias"consulte a seção.

Passo 2: Crie uma classe de armazenamento

Os usuários do Kubernetes provisionam volumes usando declarações de volume persistentes (PVCs) que especificam um "classe de armazenamento" por nome. Os detalhes ficam ocultos para os usuários, mas uma classe de storage identifica o provisionador usado para essa classe (nesse caso, Trident) e o que essa classe significa para o provisionador.

Criar uma classe de armazenamento que os usuários do Kubernetes especificarão quando quiserem um volume. A configuração da classe precisa modelar o back-end criado na etapa anterior, para que o Astra Trident a use para provisionar novos volumes.

A classe de armazenamento mais simples para começar é uma baseada no `sample-input/storage-class-csi.yaml.templ` arquivo que vem com o instalador, substituindo *`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
```

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

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

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"
        ]
      }
    }
  ]
}
```

Etapa 3: Provisione seu primeiro volume

Agora você está pronto para provisionar dinamicamente seu primeiro volume. Isso é feito criando um objeto Kubernetes "reembolso de volume persistente" (PVC).

Crie um PVC para um volume que use a classe de armazenamento que você acabou de criar.

```
`sample-input/pvc-basic-csi.yaml`Consulte para obter um exemplo. Verifique se o nome da classe de armazenamento corresponde ao que você criou.
```

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

kubectl get pvc --watch
NAME      STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS   AGE
basic     Pending
basic          1s
basic     Pending   pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   0
basic          5s
basic     Bound     pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   1Gi
RWO           basic          7s
```

Etapa 4: Montar os volumes em um pod

Agora vamos montar o volume. Vamos lançar um pod nginx que monta o PV sob `/usr/share/nginx/html`.

```
cat << EOF > task-pv-pod.yaml
kind: Pod
apiVersion: v1
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
EOF
kubectl create -f task-pv-pod.yaml
```

```
# Wait for the pod to start
kubectl get pod --watch

# Verify that the volume is mounted on /usr/share/nginx/html
kubectl exec -it task-pv-pod -- df -h /usr/share/nginx/html

# Delete the pod
kubectl delete pod task-pv-pod
```

Neste ponto, o pod (aplicativo) não existe mais, mas o volume ainda está lá. Você pode usá-lo de outro pod, se quiser.

Para eliminar o volume, elimine a reclamação:

```
kubectl delete pvc basic
```

Agora você pode fazer tarefas adicionais, como as seguintes:

- "Configurar backends adicionais."
- "Crie classes de armazenamento adicionais."

Gerenciar o Astra Trident

Atualizar o Astra Trident

O Astra Trident segue uma cadência de lançamento trimestral, entregando quatro grandes lançamentos todos os anos. Cada nova versão é construída sobre as versões anteriores, fornecendo novos recursos e melhorias de desempenho, bem como correções de bugs e melhorias. Recomendamos que você atualize pelo menos uma vez por ano para aproveitar os novos recursos do Astra Trident.

Atualizar para uma versão com cinco versões anteriores exigirá que você execute uma atualização em várias etapas.

Determine a versão para a qual atualizar

- Você pode atualizar para a `YY.MM` versão a partir `YY-1.MM` da versão e quaisquer versões intermédias. Por exemplo, você pode executar uma atualização direta para 20,07 a partir de 19,07 e posterior (incluindo lançamentos de pontos, como 19.07.1).
- Se você tiver uma versão anterior, você deve executar uma atualização em várias etapas. Isso requer que você primeiro atualize para a versão mais recente que se encaixa na janela de quatro versões. Por exemplo, se você estiver executando o 18,07 e quiser atualizar para a versão 20,07, siga o processo de atualização em várias etapas conforme indicado abaixo:
  - Primeiro upgrade de 18,07 para 19,07. Consulte a documentação da respetiva versão para obter instruções específicas para atualização.
  - Em seguida, atualize de 19,07 para 20,07.

Todas as atualizações para as versões 19,04 e anteriores exigem a migração dos metadados do Astra Trident `etcd` para objetos CRD. Verifique a documentação da versão para entender como a atualização funciona.

Ao atualizar, é importante que você forneça `parameter.fsType` o `StorageClasses` Astra Trident usado. Você pode excluir e recriar `StorageClasses` sem interromper volumes pré-existentes. Este é um **requisito** para aplicação de "contextos de segurança" volumes SAN. O "entrada de amostra" diretório contém exemplos, tais como[`storage-class-basic.yaml.templ`:[`storage-class-bronze-default.yaml` Para obter mais informações, "Problemas conhecidos"consulte .

Qual caminho de atualização devo escolher?

Você pode atualizar usando um dos seguintes caminhos:

- Utilizando o operador Trident.
- Usando `tridentctl`o .

O CSI volume Snapshots agora é um recurso que é o GA, começando com o Kubernetes 1,20. Ao atualizar o Astra Trident, todos os CRS e CRDs de instantâneo alfa anteriores (classes de instantâneo de volume, instantâneos de volume e conteúdo de instantâneo de volume) devem ser removidos antes que a atualização seja executada. "este blog"Consulte para compreender os passos envolvidos na migração de instantâneos alfa para a especificação beta/GA.

Você pode usar o operador Trident para atualizar se as seguintes condições forem atendidas:

- Você está executando o CSI Trident (19,07 e posterior).
- Você tem uma versão Trident baseada em CRD (19,07 e posterior).
- Você não está** executando uma instalação personalizada (usando YAMLs personalizados).

> Não utilize o operador para atualizar o Trident se estiver a utilizar uma `etcd` versão Trident baseada em -(19,04 ou anterior).

Se não pretender utilizar o operador ou se tiver uma instalação personalizada que não possa ser suportada pelo operador, pode atualizar utilizando `tridentctl`o . Este é o método preferido de atualizações para as versões 19,04 e anteriores do Trident.

Alterações ao operador

O lançamento de 21,01 do Astra Trident introduz algumas mudanças importantes na arquitetura do operador, nomeadamente as seguintes:

- O operador está agora **com escopo de cluster**. Instâncias anteriores do operador Trident (versões 20,04 a 20,10) eram **com escopo de namespace**. Um operador com escopo de cluster é vantajoso pelas seguintes razões:
  - Responsabilidade dos recursos: A operadora agora gerencia recursos associados à instalação do Astra Trident no nível do cluster. Como parte da instalação do Astra Trident, o operador cria e mantém vários recursos `ownerReferences` usando o . Manter `ownerReferences` recursos com escopo de cluster pode lançar erros em certos distribuidores do Kubernetes, como o OpenShift. Isso é atenuado com um operador com escopo de cluster. Para auto-cura e correção de recursos do Trident, este é um requisito essencial.
  - Limpeza durante a desinstalação: Uma remoção completa do Astra Trident exigiria que todos os recursos associados fossem excluídos. Um operador com escopo de namespace pode ter problemas com a remoção de recursos com escopo de cluster (como o clusterRole, ClusterRoleBinding e PodSecurityPolicy) e levar a uma limpeza incompleta. Um operador com escopo de cluster elimina esse problema. Os usuários podem desinstalar completamente o Astra Trident e instalar novamente, se necessário.
- `TridentProvisioner` Agora é substituído pelo `TridentOrchestrator` como recurso personalizado usado para instalar e gerenciar o Astra Trident. Além disso, um novo campo é introduzido à `TridentOrchestrator` especificação. Os usuários podem especificar que o namespace Trident deve ser instalado/atualizado usando o `spec.namespace` campo. Você pode dar uma olhada em um exemplo "aqui".

Encontre mais informações

- "Atualize utilizando o operador Trident"
  *

Atualize com o operador

Você pode atualizar facilmente uma instalação existente do Astra Trident usando o operador.

O que você vai precisar

Para atualizar usando o operador, as seguintes condições devem ser atendidas:

- Você deve ter uma instalação do Astra Trident baseada em CSI. Para verificar se você está executando o CSI Trident, examine os pods no namespace do Trident. Se eles seguirem o `trident-csi-*` padrão de nomenclatura, você está executando o CSI Trident.
- Você deve ter uma instalação Trident baseada em CRD. Isso representa todas as versões do 19,07 e posteriores. Se você tiver uma instalação baseada em CSI, provavelmente terá uma instalação baseada em CRD.
- Se tiver desinstalado o CSI Trident e os metadados da instalação persistirem, pode atualizar utilizando o operador.
- Somente uma instalação do Astra Trident deve existir em todos os namespaces em um determinado cluster do Kubernetes.
- Você deve estar usando um cluster do Kubernetes que executa "versão 1,18 - 1,24"o .
- Se CRDs de snapshot alfa estiverem presentes, você deve removê-los com `tridentctl obliviate alpha-snapshot-crd`. Isso exclui as CRDs para a especificação do instantâneo alfa. Para obter instantâneos existentes que devem ser excluídos/migrados, "este blog"consulte .

> Ao atualizar o Trident usando o operador na Plataforma de contêiner OpenShift, você deve atualizar para o Trident 21.01.1 ou posterior. O operador Trident lançado com 21.01.0 contém um problema conhecido que foi corrigido no 21.01.1. Para obter mais detalhes, consulte "Detalhes do problema no GitHub" .

Atualizar uma instalação de operador com escopo de cluster

Para atualizar de **Trident 21,01 e posterior**, aqui está o conjunto de etapas a serem seguidas.

Passos

1. Exclua o operador Trident que foi usado para instalar a instância atual do Astra Trident. Por exemplo, se você estiver atualizando do 21,01, execute o seguinte comando:

```
kubectl delete -f 21.01/trident-installer/deploy/bundle.yaml -n trident
```

2. (Opcional) se você quiser modificar os parâmetros de instalação, edite o `TridentOrchestrator` objeto que você criou ao instalar o Trident. Isso pode incluir alterações, como modificar a imagem Trident personalizada, Registro de imagem privado para extrair imagens de contentor, ativar logs de depuração ou especificar segredos de recebimento de imagens.
3. Instale o Astra Trident usando o `bundle.yaml` arquivo que configura o operador Trident para a nova versão. Execute o seguinte comando:

```
kubectl create -f 21.10.0/trident-installer/deploy/bundle.yaml -n
trident
```

Como parte desta etapa, o operador do 21.10.0 Trident identificará uma instalação existente do Astra Trident e a atualizará para a mesma versão do operador.

Atualize uma instalação de operador com escopo de namespace

Para atualizar de uma instância do Astra Trident instalada usando o operador com escopo de namespace (versões 20,07 a 20,10), aqui está o conjunto de etapas a serem seguidas:

Passos

1. Verifique o status da instalação existente do Trident. Para fazer isso, verifique o **Status** de `TridentProvisioner`. O estado deve ser `Installed`.

```
$ kubectl describe tprov trident -n trident | grep Message: -A 3
Message:  Trident installed
Status:   Installed
Version:  v20.10.1
```

> Se o status for exibido `Updating`, certifique-se de resolvê-lo antes de prosseguir. Para obter uma lista de valores de estado possíveis, "aqui"consulte .

2. Crie o `TridentOrchestrator` CRD usando o manifesto fornecido com o instalador do Trident.

```
# Download the release required [21.01]
$ mkdir 21.07.1
$ cd 21.07.1
$ wget
https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
$ tar -xf trident-installer-21.07.1.tar.gz
$ cd trident-installer
$ kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. Exclua o operador com escopo de namespace usando seu manifesto. Para concluir esta etapa, você precisa do `bundle.yaml` arquivo usado para implantar o operador com escopo de namespace. Pode `bundle.yaml` obter a partir do "Repositório Trident". Certifique-se de usar a ramificação apropriada.

> Você deve fazer as alterações necessárias nos parâmetros de instalação do Trident (por exemplo, alterando os valores para `tridentImage` `autosupportImage` , , repositório de imagens privado e fornecendo `imagePullSecrets`) depois de excluir o operador com escopo de namespace e antes de instalar o operador com escopo de cluster. Para obter uma lista completa de parâmetros que podem ser atualizados, consulte "lista de parâmetros".

```
#Ensure you are in the right directory
$ pwd
$ /root/20.10.1/trident-installer

#Delete the namespace-scoped operator
$ kubectl delete -f deploy/bundle.yaml
serviceaccount "trident-operator" deleted
clusterrole.rbac.authorization.k8s.io "trident-operator" deleted
clusterrolebinding.rbac.authorization.k8s.io "trident-operator" deleted
deployment.apps "trident-operator" deleted
podsecuritypolicy.policy "tridentoperatorpods" deleted

#Confirm the Trident operator was removed
$ kubectl get all -n trident
NAME                                READY   STATUS    RESTARTS   AGE
pod/trident-csi-68d979fb85-dsrmn    6/6     Running   12         99d
pod/trident-csi-8jfhf               2/2     Running   6          105d
pod/trident-csi-jtnjz               2/2     Running   6          105d
pod/trident-csi-lcxvh               2/2     Running   8          105d

NAME                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
service/trident-csi   ClusterIP   10.108.174.125   <none>
34571/TCP,9220/TCP   105d

NAME                           DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR                                        AGE
daemonset.apps/trident-csi   3         3         3       3            3
kubernetes.io/arch=amd64,kubernetes.io/os=linux   105d

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/trident-csi   1/1     1            1           105d

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/trident-csi-68d979fb85   1         1         1
105d
```

Nesta fase, o `trident-operator-xxxxxxxxxx-xxxxx` pod é excluído.

4. (Opcional) se os parâmetros de instalação precisarem ser modificados, atualize a `TridentProvisioner` especificação. Essas alterações podem ser alterações, como modificar o Registro de imagens privadas para extrair imagens de contentor, ativar logs de depuração ou especificar segredos de recebimento de imagens.

```
$  kubectl patch tprov <trident-provisioner-name> -n <trident-namespace>
--type=merge -p '{"spec":{"debug":true}}'
```

5. Instale o operador com escopo de cluster.

> A instalação do operador com escopo de cluster inicia a migração `TridentProvisioner` de objetos para `TridentOrchestrator` objetos, exclui `TridentProvisioner` objetos e `tridentprovisioner` CRD e atualiza o Astra Trident para a versão do operador com escopo de cluster que está sendo usado. No exemplo a seguir, o Trident é atualizado para 21.07.1.

> A atualização do Astra Trident usando o operador com escopo de cluster resulta na migração de `tridentProvisioner` para um `tridentOrchestrator` objeto com o mesmo nome. Este procedimento é automaticamente gerido pelo operador. A atualização também terá o Astra Trident instalado no mesmo namespace que antes.

```
#Ensure you are in the correct directory
$ pwd
$ /root/21.07.1/trident-installer

#Install the cluster-scoped operator in the **same namespace**
$ kubectl create -f deploy/bundle.yaml
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#All tridentProvisioners will be removed, including the CRD itself
$ kubectl get tprov -n trident
Error from server (NotFound): Unable to list "trident.netapp.io/v1,
Resource=tridentprovisioners": the server could not find the requested
resource (get tridentprovisioners.trident.netapp.io)

#tridentProvisioners are replaced by tridentOrchestrator
$ kubectl get torc
NAME      AGE
trident   13s

#Examine Trident pods in the namespace
$ kubectl get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m41s
trident-csi-xrst8                   2/2     Running   0          1m41s
trident-operator-5574dbbc68-nthjv   1/1     Running   0          1m52s

#Confirm Trident has been updated to the desired version
$ kubectl describe torc trident | grep Message -A 3
Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v21.07.1
```

Atualize uma instalação de operador baseada em Helm

Execute as etapas a seguir para atualizar uma instalação do operador baseada em Helm.

Passos

1. Baixe o mais recente lançamento do Astra Trident.
2. Use o `helm upgrade` comando. Veja o exemplo a seguir:

```
$ helm upgrade <name> trident-operator-21.07.1.tgz
```

onde `trident-operator-21.07.1.tgz` reflete a versão para a qual você deseja atualizar.

3. Execute `helm list` para verificar se o gráfico e a versão do aplicativo foram atualizados.

> Para passar dados de configuração durante a atualização, use `--set`o .

Por exemplo, para alterar o valor padrão `tridentDebug` do , execute o seguinte comando:

```
$ helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
tridentDebug=true
```

Se você executar `$ tridentctl logs`o , você poderá ver as mensagens de depuração.

> Se você definir qualquer opção não padrão durante a instalação inicial, certifique-se de que as opções estão incluídas no comando upgrade, ou então, os valores serão redefinidos para seus padrões.

Atualize a partir de uma instalação que não seja do operador

Se você tiver uma instância do CSI Trident que atenda aos pré-requisitos listados acima, poderá atualizar para a versão mais recente do operador Trident.

Passos

1. Baixe o mais recente lançamento do Astra Trident.

```
# Download the release required [21.07.1]
$ mkdir 21.07.1
$ cd 21.07.1
$ wget
https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
$ tar -xf trident-installer-21.07.1.tar.gz
$ cd trident-installer
```

2. Crie o `tridentorchestrator` CRD a partir do manifesto.

```
$ kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. Implante o operador.

```
#Install the cluster-scoped operator in the **same namespace**
$ kubectl create -f deploy/bundle.yaml
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          150d
trident-csi-xrst8                   2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv   1/1     Running   0          1m30s
```

4. Crie `TridentOrchestrator` um CR para a instalação do Astra Trident.

```
#Create a tridentOrchestrator to initate a Trident install
$ cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident

$ kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s

#Confirm Trident was upgraded to the desired version
$ kubectl describe torc trident | grep Message -A 3
Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v21.07.1
```

Os backends e PVCs existentes estão disponíveis automaticamente.

Atualize com o tridentctl

Você pode atualizar facilmente uma instalação existente do Astra Trident usando `tridentctl`o .

Considerações

Ao atualizar para a versão mais recente do Astra Trident, considere o seguinte:

- A partir do Trident 20,01, apenas a versão beta do "instantâneos de volume" é suportada. Os administradores do Kubernetes devem cuidar para fazer backup ou converter com segurança os objetos de snapshot alfa em beta para reter os snapshots alfa legados.
- A versão beta de snapshots de volume introduz um conjunto modificado de CRDs e um controlador de snapshot, que devem ser configurados antes de instalar o Astra Trident.

"Este blog" discute as etapas envolvidas na migração de snapshots de volume alfa para o formato beta.

Sobre esta tarefa

Desinstalar e reinstalar o Astra Trident funciona como uma atualização. Quando você desinstalar o Trident, a reivindicação de volume persistente (PVC) e o volume persistente (PV) usados pela implantação do Astra Trident não são excluídos. Os PVS que já tiverem sido provisionados permanecerão disponíveis enquanto o Astra Trident estiver offline, e o Astra Trident provisionará volumes para quaisquer PVCs que forem criados nesse período, uma vez que estiverem novamente online.

Ao atualizar o Astra Trident, não interrompa o processo de atualização. Certifique-se de que o instalador é executado até a conclusão.

Próximos passos após a atualização

Para fazer uso do rico conjunto de recursos que estão disponíveis em versões mais recentes do Trident (como snapshots de volume sob demanda), você pode atualizar os volumes usando o `tridentctl upgrade` comando.

Se houver volumes legados, você deve atualizá-los de um tipo NFS/iSCSI para o tipo CSI para poder usar o conjunto completo de novos recursos no Astra Trident. Um PV legado que foi provisionado pelo Trident suporta o conjunto tradicional de recursos.

Considere o seguinte ao decidir atualizar volumes para o tipo CSI:

- Talvez você não precise atualizar todos os volumes. Os volumes criados anteriormente continuarão a ser acessíveis e a funcionar normalmente.
- Um PV pode ser montado como parte de um deployment/StatefulSet ao atualizar. Não é necessário derrubar o deployment/StatefulSet.
- Você **não pode** anexar um PV a um pod autônomo ao atualizar. Você deve desligar o pod antes de atualizar o volume.
- Você pode atualizar apenas um volume que esteja vinculado a um PVC. Os volumes que não estão vinculados a PVCs devem ser removidos e importados antes da atualização.

Exemplo de atualização de volume

Aqui está um exemplo que mostra como uma atualização de volume é realizada.

1. Execute `kubectl get pv` para listar os PVS.

```
$ kubectl get pv
NAME                        CAPACITY     ACCESS MODES   RECLAIM POLICY
STATUS   CLAIM                   STORAGECLASS    REASON   AGE
default-pvc-1-a8475         1073741824   RWO            Delete
Bound    default/pvc-1          standard                19h
default-pvc-2-a8486         1073741824   RWO            Delete
Bound    default/pvc-2          standard                19h
default-pvc-3-a849e         1073741824   RWO            Delete
Bound    default/pvc-3          standard                19h
default-pvc-4-a84de         1073741824   RWO            Delete
Bound    default/pvc-4          standard                19h
trident                     2Gi          RWO            Retain
Bound    trident/trident                                19h
```

Atualmente, existem quatro PVS que foram criados pelo Trident 20,07, usando o `netapp.io/trident` provisionador.

2. Corra `kubectl describe pv` para obter os detalhes do PV.

```
$ kubectl describe pv default-pvc-2-a8486

Name:            default-pvc-2-a8486
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: netapp.io/trident
                 volume.beta.kubernetes.io/storage-class: standard
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    standard
Status:          Bound
Claim:           default/pvc-2
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        1073741824
Node Affinity:   <none>
Message:
Source:
    Type:      NFS (an NFS mount that lasts the lifetime of a pod)
    Server:    10.xx.xx.xx
    Path:      /trid_1907_alpha_default_pvc_2_a8486
    ReadOnly:  false
```

O PV foi criado usando o `netapp.io/trident` provisionador e é do tipo NFS. Para suportar todos os novos recursos fornecidos pelo Astra Trident, este PV deve ser atualizado para o tipo CSI.

3. Execute o `tridenctl upgrade volume <name-of-trident-volume>` comando para atualizar um volume legado Astra Trident para a especificação CSI.

```
$ ./tridentctl get volumes -n trident
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
|            NAME     |  SIZE   | STORAGE CLASS | PROTOCOL |
BACKEND UUID             | STATE  | MANAGED |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-3-a849e | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-1-a8475 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-4-a84de | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+

$ ./tridentctl upgrade volume default-pvc-2-a8486 -n trident
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
|            NAME     |  SIZE   | STORAGE CLASS | PROTOCOL |
BACKEND UUID             | STATE  | MANAGED |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
```

4. Execute um `kubectl describe pv` para verificar se o volume é um volume CSI.

```
$ kubectl describe pv default-pvc-2-a8486
Name:            default-pvc-2-a8486
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: csi.trident.netapp.io
                 volume.beta.kubernetes.io/storage-class: standard
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    standard
Status:          Bound
Claim:           default/pvc-2
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        1073741824
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume
source)
    Driver:            csi.trident.netapp.io
    VolumeHandle:      default-pvc-2-a8486
    ReadOnly:          false
    VolumeAttributes:      backendUUID=c5a6f6a4-b052-423b-80d4-
8fb491a14a22

internalName=trid_1907_alpha_default_pvc_2_a8486
                           name=default-pvc-2-a8486
                           protocol=file
Events:                <none>
```

Dessa forma, você pode atualizar volumes do tipo NFS/iSCSI criados pelo Astra Trident para o tipo CSI, por volume.

Desinstale o Astra Trident

Dependendo de como o Astra Trident é instalado, há várias opções para desinstalá-lo.

Desinstalar usando Helm

Se você instalou o Astra Trident usando o Helm, você pode desinstalá-lo usando `helm uninstall`o .

```
#List the Helm release corresponding to the Astra Trident install.
$ helm ls -n trident
NAME          NAMESPACE       REVISION        UPDATED
STATUS          CHART                           APP VERSION
trident       trident         1               2021-04-20
00:26:42.417764794 +0000 UTC deployed        trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
$ helm uninstall trident -n trident
release "trident" uninstalled
```

Desinstale utilizando o operador Trident

Se você instalou o Astra Trident usando o operador, você pode desinstalá-lo fazendo um dos seguintes procedimentos:

- **Editar `TridentOrchestrator` para definir o sinalizador de desinstalação:** você pode editar `TridentOrchestrator` e definir `spec.uninstall=true`. Edite o `TridentOrchestrator` CR e defina a `uninstall` bandeira como mostrado abaixo:

  ```
  $  kubectl patch torc <trident-orchestrator-name> --type=merge -p
  '{"spec":{"uninstall":true}}'
  ```

  Quando o `uninstall` sinalizador está definido como `true`, o operador Trident desinstala o Trident, mas não remove o próprio TridentOrchestrator. Você deve limpar o TridentOrchestrator e criar um novo se quiser instalar o Trident novamente.

- **Excluir `TridentOrchestrator`:** removendo o `TridentOrchestrator` CR que foi usado para implantar o Astra Trident, você instrui o operador a desinstalar o Trident. O operador processa a remoção `TridentOrchestrator` e remove a implantação do Astra Trident e o daemonset, excluindo os pods do Trident que ele criou como parte da instalação. Para remover completamente o Astra Trident (incluindo as CRDs que ele cria) e limpar efetivamente a ardósia limpa, você pode editar `TridentOrchestrator` para passar a `wipeout` opção. Veja o exemplo a seguir:

```
$ kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

Isso desinstala completamente o Astra Trident e limpa todos os metadados relacionados aos back-ends e volumes que ele gerencia. As instalações subsequentes são tratadas como instalações frescas.

> Você só deve considerar limpar as CRDs ao executar uma desinstalação completa. Isso não pode ser desfeito. **Não limpe as CRDs a menos que você esteja olhando para começar de novo e criar uma nova instalação do Astra Trident**.

Desinstalar usando `tridentctl`

Execute o `uninstall` comando `tridentctl` da seguinte forma para remover todos os recursos associados ao Astra Trident, exceto para os CRDs e objetos relacionados, facilitando assim a execução do instalador novamente para atualizar para uma versão mais recente.

```
./tridentctl uninstall -n <namespace>
```

Para realizar uma remoção completa do Astra Trident, você deve remover os localizadores das CRDs criadas pelo Astra Trident e excluir os CRDs.

Downgrade Astra Trident

Saiba mais sobre os passos envolvidos na redução para uma versão anterior do Astra Trident.

Você pode considerar downgrade por vários motivos, como o seguinte:

- Planejamento de contingência
- Correção imediata para bugs observados como resultado de uma atualização
- Problemas de dependência, atualizações sem sucesso e incompletas

Quando fazer downgrade

Você deve considerar um downgrade ao mudar para uma versão do Astra Trident que usa CRDs. Como o Astra Trident agora usa CRDs para manter o estado, todas as entidades de storage criadas (backends, classes de storage, PV e snapshots de volume) têm objetos CRD associados em vez de dados gravados `trident` no PV (usados pela versão instalada anterior do Astra Trident). PVS, backends e classes de armazenamento recém-criados são todos mantidos como objetos CRD. Se você precisar fazer o downgrade, isso só deve ser tentado para uma versão do Astra Trident que seja executada usando CRDs (19,07 e posterior). Isso garante que todas as operações realizadas no atual lançamento do Astra Trident sejam visíveis após o downgrade ocorrer.

Quando não para downgrade

Você não deve fazer o downgrade para uma versão do Trident que usa `etcd` para manter o estado (19,04 e anterior). Todas as operações executadas com o lançamento atual do Astra Trident não se refletem após o downgrade. PVS recém-criados não são utilizáveis ao voltar para uma versão anterior. As alterações feitas em objetos como backends, PVS, classes de armazenamento e snapshots de volume (criados/atualizados/excluídos) não são visíveis para o Astra Trident ao voltar para uma versão anterior. Voltar para uma versão anterior não interrompe o acesso para PVS que já foram criados usando a versão mais antiga, a menos que tenham sido atualizados.

Processo de downgrade quando o Astra Trident é instalado usando o operador

Para instalações realizadas usando o Operador Trident, o processo de downgrade é diferente e não requer o uso `tridentctl`do .

Para instalações feitas usando o operador Trident, o Astra Trident pode ser rebaixado para uma das seguintes opções:

- Uma versão que é instalada usando o operador com escopo de namespace (20,07 - 20,10).
- Uma versão que é instalada usando o operador com escopo de cluster (21,01 e posterior).

Downgrade para operador com escopo de cluster

Para fazer o downgrade do Astra Trident para uma versão que usa o operador com escopo de cluster, siga as etapas mencionadas abaixo.

Passos

1. "Desinstale o Astra Trident". **Não use os CRDs a menos que você queira remover completamente uma instalação existente.**
2. Exclua o operador com escopo de cluster. Para fazer isso, você precisará do manifesto usado para implantar o operador. Você pode obtê-lo a partir do "Trident repo". Certifique-se de mudar para a filial pretendida.
3. Continue a fazer o downgrade instalando a versão desejada do Astra Trident. Siga a documentação para a versão desejada.

Downgrade para operador com escopo de namespace

Esta seção resume as etapas envolvidas na redução para uma versão do Astra Trident que cai no intervalo de 20,07 a 20,10, que será instalada usando o operador com escopo de namespace.

Passos

1. "Desinstale o Astra Trident". **Não use os CRDs a menos que você queira remover completamente uma instalação existente.** Certifique-se de que o `tridentorchestrator` é eliminado.

```
#Check to see if there are any tridentorchestrators present
$ kubectl get torc
NAME        AGE
trident     20h

#Looks like there is a tridentorchestrator that needs deleting
$ kubectl delete torc trident
tridentorchestrator.trident.netapp.io "trident" deleted
```

2. Exclua o operador com escopo de cluster. Para fazer isso, você precisará do manifesto usado para implantar o operador. Você pode obtê-lo aqui a partir do "Trident repo". Certifique-se de mudar para a filial pretendida.
3. Eliminar o `tridentorchestrator` CRD.

```
#Check to see if ``tridentorchestrators.trident.netapp.io`` CRD is
present and delete it.
$ kubectl get crd tridentorchestrators.trident.netapp.io
NAME                                        CREATED AT
tridentorchestrators.trident.netapp.io   2021-01-21T21:11:37Z
$ kubectl delete crd tridentorchestrators.trident.netapp.io
customresourcedefinition.apiextensions.k8s.io
"tridentorchestrators.trident.netapp.io" deleted
```

O Astra Trident foi desinstalado.

4. Continue a baixar instalando a versão desejada. Siga a documentação para a versão desejada.

Downgrade usando Helm

Para fazer o downgrade, use o `helm rollback` comando. Veja o exemplo a seguir:

```
$ helm rollback trident [revision #]
```

Processo de downgrade quando o Astra Trident é instalado usando `tridentctl`

Se você instalou o Astra Trident usando `tridentctl`o , o processo de downgrade envolve as etapas a seguir. Esta sequência orienta você pelo processo de downgrade para passar do Astra Trident 21,07 para 20,07.

> Antes de começar o downgrade, você deve tirar um snapshot dos clusters do Kubernetes `etcd`. Isso permite fazer backup do estado atual das CRDs do Astra Trident.

Passos

1. Certifique-se de que o Trident esteja instalado `tridentctl` usando o . Se você não tiver certeza sobre como o Astra Trident está instalado, execute este teste simples:
   a. Liste os pods presentes no namespace Trident.
   b. Identifique a versão do Astra Trident em execução no cluster. Você pode `tridentctl` usar ou dar uma olhada na imagem usada nos pods do Trident.
   c. Se **não vir** a `tridentOrchestrator`, (ou) a `tridentprovisioner`, (ou) um pod chamado `trident-operator-xxxxxxxxxx-xxxxx`, o Astra Trident **está instalado** com `tridentctl`.
2. Desinstale o Astra Trident com o binário existente `tridentctl`. Neste caso, você irá desinstalar com o binário 21,07.

```
$ tridentctl version -n trident
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.0        | 21.07.0        |
+----------------+----------------+

$ tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted Trident daemonset.
INFO Deleted Trident service.
INFO Deleted Trident secret.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Deleted pod security policy.
podSecurityPolicy=tridentpods
INFO The uninstaller did not delete Trident's namespace in case it is
going to be reused.
INFO Trident uninstallation succeeded.
```

3. Depois que isso for concluído, obtenha o binário Trident para a versão desejada (neste exemplo, 20,07) e use-o para instalar o Astra Trident. Você pode gerar YAMLs personalizados para um"instalação personalizada", se necessário.

```
$ cd 20.07/trident-installer/
$ ./tridentctl install -n trident-ns
INFO Created installer service account.
serviceaccount=trident-installer
INFO Created installer cluster role.              clusterrole=trident-
installer
INFO Created installer cluster role binding.
clusterrolebinding=trident-installer
INFO Created installer configmap.                 configmap=trident-
installer
...
...
INFO Deleted installer cluster role binding.
INFO Deleted installer cluster role.
INFO Deleted installer service account.
```

O processo de downgrade está concluído.

Use o Astra Trident

Configurar backends

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

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

Configurar um back-end do Azure NetApp Files

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

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

O que você vai precisar

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

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

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

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

Sobre esta tarefa

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

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

Opções de configuração de back-end

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

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

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

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

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

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

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

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

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

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

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

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

Exemplo 1: Configuração mínima

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

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

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
    "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
    "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
    "clientSecret": "SECRET",
    "location": "eastus"
}
```

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

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

```
    {
        "version": 1,
        "storageDriverName": "azure-netapp-files",
        "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
        "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
        "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
        "clientSecret": "SECRET",
        "location": "eastus",
        "serviceLevel": "Ultra",
        "capacityPools": [
            "application-group-1/account-1/ultra-1",
            "application-group-1/account-1/ultra-2"
],
    }
```

Exemplo 3: Configuração avançada

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

```
    {
        "version": 1,
        "storageDriverName": "azure-netapp-files",
        "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
        "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
        "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
        "clientSecret": "SECRET",
        "location": "eastus",
        "serviceLevel": "Ultra",
        "capacityPools": [
            "application-group-1/account-1/ultra-1",
            "application-group-1/account-1/ultra-2"
],
        "virtualNetwork": "my-virtual-network",
        "subnet": "my-subnet",
        "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
        "limitVolumeSize": "500Gi",
        "defaults": {
            "exportRule": "10.0.0.0/24,10.0.1.0/24,10.0.2.100",
            "snapshotDir": "true",
            "size": "200Gi",
            "unixPermissions": "0777"
=======
        }
    }
```

Exemplo 4: Configuração do pool de armazenamento virtual

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

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
    "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
    "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
    "clientSecret": "SECRET",
    "location": "eastus",
    "resourceGroups": ["application-group-1"],
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
    "labels": {
        "cloud": "azure"
    },
    "location": "eastus",

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

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

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

O que se segue?

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

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

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

```
tridentctl logs
```

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

Configurar um back-end do CVS para GCP

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

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

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

O que você vai precisar

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

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

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

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

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

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

Opções de configuração de back-end

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Exemplo 1: Configuração mínima

Esta é a configuração mínima absoluta de back-end.

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

Exemplo 2: Configuração do tipo de serviço CVS básico

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

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

Exemplo 3: Configuração de nível de serviço único

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

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

Exemplo 4: Configuração do pool de armazenamento virtual

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

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

```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "apiRegion": "us-west2",
    "apiKey": {
        "type": "service_account",
        "private_key_id": "<id_value>",
        "private_key": "
        -----BEGIN PRIVATE KEY-----
        <key_value>
        -----END PRIVATE KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    },
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",

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

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

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

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

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

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

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

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

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

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

O que se segue?

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

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

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

```
tridentctl logs
```

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

Configurar um back-end NetApp HCI ou SolidFire

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

O que você vai precisar

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

O que você precisa saber

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

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

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

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

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

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

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

Opções de configuração de back-end

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

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do controlador de armazenamento	Sempre "SolidFire-san"
`backendName`	Nome personalizado ou back-end de storage	Endereço IP "SolidFire_" e armazenamento (iSCSI)
`Endpoint`	MVIP para o cluster SolidFire com credenciais de locatário	
`SVIP`	Porta e endereço IP de armazenamento (iSCSI)	
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes.	""
`TenantName`	Nome do locatário a utilizar (criado se não for encontrado)	

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

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

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

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

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

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

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

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

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

```
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://<user>:<password>@<mvip>/json-rpc/8.0",
    "SVIP": "<svip>:3260",
    "TenantName": "<tenant>",
    "UseCHAP": true,
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS": 2000,
"burstIOPS": 4000}},
             {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS": 6000,
"burstIOPS": 8000}},
             {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS": 8000,
"burstIOPS": 10000}}],

    "type": "Silver",
    "labels":{"store":"solidfire", "k8scluster": "dev-1-cluster"},
    "region": "us-east-1",

    "storage": [
        {
            "labels":{"performance":"gold", "cost":"4"},
            "zone":"us-east-1a",
            "type":"Gold"
        },
        {
            "labels":{"performance":"silver", "cost":"3"},
            "zone":"us-east-1b",
            "type":"Silver"
        },
        {
            "labels":{"performance":"bronze", "cost":"2"},
            "zone":"us-east-1c",
            "type":"Bronze"
        },
        {
            "labels":{"performance":"silver", "cost":"1"},
            "zone":"us-east-1d"
        }
    ]
}
```

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

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

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

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

Encontre mais informações

- "Grupos de acesso de volume"

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

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

- "Preparação"
- "Configuração e exemplos"

Permissões do usuário

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

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

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

Preparação

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

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

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

Autenticação

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

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

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

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

Ative a autenticação baseada em credenciais

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

Uma definição de backend de exemplo será assim:

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

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

Ativar autenticação baseada em certificado

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

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

Um fluxo de trabalho típico envolve as etapas a seguir.

Passos

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Atualizar métodos de autenticação ou girar credenciais

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

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

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

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

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

Especifique grupos

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

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

nós do Kubernetes ao grupo pré-existente.

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

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

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

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

Autentique conexões com CHAP bidirecional

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

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

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

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

Como funciona

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

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

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

Gire credenciais e atualize os backends

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

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

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

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

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

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

Opções de configuração e exemplos

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

Opções de configuração de back-end

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

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

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

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

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

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

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

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

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

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

Os backends também podem ter grupos atualizados após a criação:

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

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

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

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

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

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

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

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

Opções de configuração de back-end para volumes de provisionamento

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

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

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

Aqui está um exemplo com padrões definidos:

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

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

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

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

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

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

Exemplos mínimos de configuração

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

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

`ontap-san` driver com autenticação baseada em certificado

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

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

`ontap-san` Driver com CHAP bidirecional

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

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

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

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

Exemplos de backends com pools de armazenamento virtual

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

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

```
```

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

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

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

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

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

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

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

Mapeie os backends para StorageClasses

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

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

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

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

Configurar um back-end com drivers nas ONTAP

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

- "Preparação"
- "Configuração e exemplos"

Permissões do usuário

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

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

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

Preparação

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

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

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

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

Autenticação

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

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

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

end.

Ative a autenticação baseada em credenciais

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

Uma definição de backend de exemplo será assim:

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

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

Ativar autenticação baseada em certificado

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

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

Um fluxo de trabalho típico envolve as etapas a seguir.

Passos

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Atualizar métodos de autenticação ou girar credenciais

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

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

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

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

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

Gerenciar políticas de exportação de NFS

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

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

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

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

Gerencie dinamicamente políticas de exportação

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

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

Exemplo

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Opções de configuração e exemplos

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

Opções de configuração de back-end

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Opções de configuração de back-end para volumes de provisionamento

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

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

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

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

Aqui está um exemplo com padrões definidos:

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

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

O cálculo é o seguinte:

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

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

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

Exemplos mínimos de configuração

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

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

`ontap-nas` driver com autenticação baseada em certificado

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

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

`ontap-nas` driver com política de exportação automática

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

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

`ontap-nas-flexgroup` condutor

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

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

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

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

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

Exemplos de backends com pools de armazenamento virtual

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

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

`ontap-nas` **condutor**

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

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

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

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

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

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

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

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

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

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

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

Mapeie os backends para StorageClasses

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

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

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

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

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

Use o Astra Trident com o Amazon FSX para NetApp ONTAP

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

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

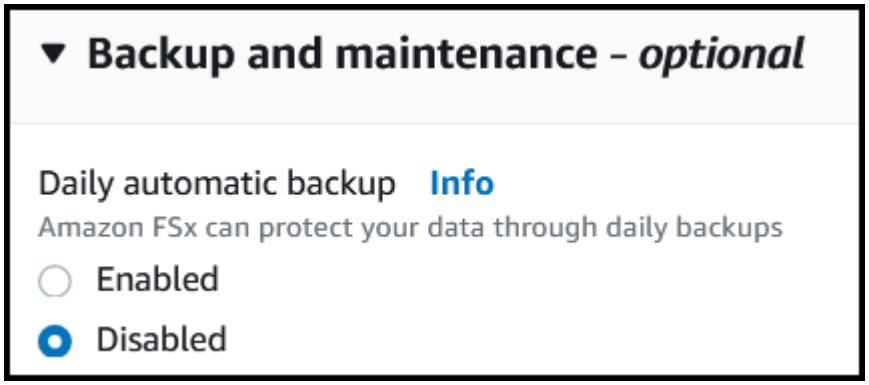
Usando o Astra Trident com o Amazon FSX for NetApp ONTAP, você pode garantir que os clusters do Kubernetes executados no Amazon Elastic Kubernetes Service (EKS) provisionem volumes persistentes de bloco e arquivo com o respaldo do do ONTAP.

Criando seu sistema de arquivos do Amazon FSX for ONTAP

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

Para evitar este problema:

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



Saiba mais sobre o Astra Trident

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

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

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

Saiba mais sobre os recursos do "aqui"driver .

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

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

Drivers

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

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

Autenticação

O Astra Trident oferece dois modos de autenticação:

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

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

Para saber mais sobre autenticação, consulte estes links:

- "ONTAP nas"
- "San ONTAP"

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

O que você vai precisar

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

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

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

Passos

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

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

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

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

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

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

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

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

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

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

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

Encontre mais informações

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

Crie backends com kubectl

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

`TridentBackendConfig`

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

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

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

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

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

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

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

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

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

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

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

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

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

Visão geral dos passos

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

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

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

Etapa 1: Crie um segredo do Kubernetes

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

```
$ kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: t@Ax@7q(>
```

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

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

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

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

Passo 2: Crie o `TridentBackendConfig` CR

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

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

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

Etapa 3: Verifique o status do `TridentBackendConfig` CR

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

```
$ kubectl -n trident get tbc backend-tbc-ontap-san
NAME                     BACKEND NAME          BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san   ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success
```

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

A fase pode ter um dos seguintes valores:

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

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

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

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

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

(Opcional) passo 4: Obtenha mais detalhes

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

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

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

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

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

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

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

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

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

Execute o gerenciamento de back-end com o kubectl

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

Excluir um back-end

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

Execute o seguinte comando:

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

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

Veja os backends existentes

Execute o seguinte comando:

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

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

Atualize um back-end

Pode haver várias razões para atualizar um backend:

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

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

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

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

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

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

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

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

Execute o gerenciamento de back-end com o tridentctl

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

Crie um backend

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

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

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

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

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

Excluir um back-end

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

1. Recuperar o nome do backend:

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

2. Excluir o backend:

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

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

Veja os backends existentes

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

- Para obter um resumo, execute o seguinte comando:

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

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

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

Atualize um back-end

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

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

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

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

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

Identificar as classes de armazenamento que usam um back-end

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

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

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

Alternar entre opções de gerenciamento de back-end

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

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

Gerenciar `tridentctl` backends usando `TridentBackendConfig`

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

Isso se aplicará aos seguintes cenários:

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

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

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

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

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

Passo 0: Identifique o backend

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

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

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

$ cat ontap-nas-backend.json

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

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

```
        }
    ]
}
```

Etapa 1: Crie um segredo do Kubernetes

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

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

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

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

Passo 2: Crie um `TridentBackendConfig` CR

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

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

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

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

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

Etapa 3: Verifique o status do `TridentBackendConfig` CR

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

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

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

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

Gerenciar `TridentBackendConfig` backends usando `tridentctl`

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

Passo 0: Identifique o backend

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

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

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

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

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

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

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

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

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

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

Etapa 2: Exclua o `TridentBackendConfig` CR

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

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

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

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

Gerenciar classes de armazenamento

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

Crie uma classe de armazenamento

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

Crie uma classe de armazenamento

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

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

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

Excluir uma classe de armazenamento

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

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

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

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

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

Exibir as classes de armazenamento existentes

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

```
kubectl get storageclass
```

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

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

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

```
tridentctl get storageclass
```

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

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

Defina uma classe de armazenamento padrão

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

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

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

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

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

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

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

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

Identificar o back-end de uma classe de storage

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

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

Executar operações de volume

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

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

Use a topologia CSI

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

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

O Kubernetes oferece dois modos exclusivos de vinculação de volume:

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

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

O que você vai precisar

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

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

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

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

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

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

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

Veja como é um exemplo de definição de backend:

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

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

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

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

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

Etapa 2: Defina StorageClasses que estejam cientes da topologia

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

Veja o exemplo a seguir:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
- key: topology.kubernetes.io/zone
  values:
  - us-east1-a
  - us-east1-b
- key: topology.kubernetes.io/region
  values:
  - us-east1
parameters:
  fsType: "ext4"
```

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

Passo 3: Criar e usar um PVC

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

Veja o exemplo `spec` abaixo:

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

Criar um PVC usando este manifesto resultaria no seguinte:

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

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

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

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

Veja a seguinte saída:

```
$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
$ kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

Atualize os backends para incluir `supportedTopologies`

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

Encontre mais informações

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

Trabalhar com instantâneos

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

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

O que você vai precisar

- A criação de instantâneos de volume requer a criação de um controlador de snapshot externo e de definições personalizadas de recursos (CRDs). Essa é a responsabilidade do orquestrador do Kubernetes sendo usado (por exemplo: Kubeadm, GKE, OpenShift).

Se a sua distribuição do Kubernetes não incluir a controladora de snapshot e CRDs, você poderá implantá-los da seguinte forma.

1. Criar CRDs de instantâneos de volume.

   Para Kubernetes 1,20 e posterior, use CRDs de snapshot de v1 com componentes de snapshot de v5,0 ou superior. Para as versões 1,18 e 1,19 do Kubernetes, use o v1beta1 com componentes de snapshot v3,0.3.

v5.0 componentes

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

v3.0.3 componentes

```
$ cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-
snapshotter/v3.0.3/client/config/crd/snapshot.storage.k8s.io_volumes
napshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-
snapshotter/v3.0.3/client/config/crd/snapshot.storage.k8s.io_volumes
napshotcontents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-
snapshotter/v3.0.3/client/config/crd/snapshot.storage.k8s.io_volumes
napshots.yaml
```

2. Crie o controlador de snapshot no namespace desejado. Edite os manifestos YAML abaixo para modificar o namespace.

   Para Kubernetes 1,20 e posterior, use o v5,0 ou superior. Para as versões 1,18 e 1,19 do Kubernetes, use o v3,0.3

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

Controlador v5.0

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-5.0/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-5.0/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```

Controlador v3.0.3

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/v3.0.3/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/v3.0.3/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```

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

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

Passo 1: Configure a. `VolumeSnapshotClass`

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

```
$ cat snap-sc.yaml
#Use apiVersion v1 for Kubernetes 1.20 and above. For Kubernetes 1.18 and 1.19, use apiVersion v1beta1.
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

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

Passo 2: Crie um instantâneo de um PVC existente

```
$ cat snap.yaml
#Use apiVersion v1 for Kubernetes 1.20 and above. For Kubernetes 1.18 and
1.19, use apiVersion v1beta1.
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

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

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

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

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

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

```
$ kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
.
.
.
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
.
.
```

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

Etapa 3: Criar PVCs a partir do VolumeSnapshots

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

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

`dataSource` Mostra que o PVC deve ser criado usando um VolumeSnapshot nomeado `pvc1-snap` como a

fonte dos dados. Isso instrui o Astra Trident a criar um PVC a partir do snapshot. Depois que o PVC é criado, ele pode ser anexado a um pod e usado como qualquer outro PVC.

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

Encontre mais informações

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

Expanda volumes

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

Expanda um volume iSCSI

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

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

Visão geral

A expansão de um iSCSI PV inclui os seguintes passos:

- Editando a definição StorageClass para definir o `allowVolumeExpansion` campo como `true`.
- Editar a definição de PVC e atualizar o `spec.resources.requests.storage` para refletir o tamanho recém-desejado, que deve ser maior que o tamanho original.
- A fixação do PV deve ser fixada a um pod para que ele seja redimensionado. Existem dois cenários ao redimensionar um iSCSI PV:
  - Se o PV estiver conetado a um pod, o Astra Trident expande o volume no back-end de armazenamento, refaz o dispositivo e redimensiona o sistema de arquivos.
  - Ao tentar redimensionar um PV não anexado, o Astra Trident expande o volume no back-end de armazenamento. Depois que o PVC é ligado a um pod, o Trident refaz o dispositivo e redimensiona o sistema de arquivos. Em seguida, o Kubernetes atualiza o tamanho do PVC após a operação de expansão ter sido concluída com sucesso.

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

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

```
$ cat storageclass-ontapsan.yaml
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

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

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

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

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

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

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

Passo 3: Defina um pod que prende o PVC

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

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

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

Passo 4: Expanda o PV

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

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

Etapa 5: Validar a expansão

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

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

Expandir um volume NFS

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

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

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

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

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

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

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

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

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

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

Passo 3: Expanda o PV

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

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

Etapa 4: Validar a expansão

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

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

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

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

Importar volumes

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

Drivers que suportam importação de volume

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

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

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

Por que devo importar volumes?

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

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

Como funciona a importação?

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`ontap-san` importar

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

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

que deve ser importado.

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

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

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

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

Exemplo

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

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

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

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

`element` importar

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

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

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

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

`gcp-cvs` importar

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

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

```
$ tridentctl import volume gcpcvs_YEppr adroit-jolly-swift -f <path-to-
pvc-file> -n trident

+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | 93 GiB | gcp-storage   | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

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

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

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

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

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

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

Prepare o nó de trabalho

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

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

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

Volumes NFS

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

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

Volumes iSCSI

Considere o seguinte ao usar volumes iSCSI:

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

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

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

Protocolo	Sistema operacional	Comandos

ISCSI	RHEL/CentOS	1. Instale os seguintes pacotes de sistema:
		
		`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`

Protocolo	Sistema operacional	Comandos

ISCSI	Ubuntu/Debian	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` `defaults` em .
		
		5. Certifique-se de que `open-iscsi` e `multipath-tools` estão ativados e em execução:
		
		`sudo systemctl status multipath-tools`

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

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

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

Preparação automática do nó de trabalho

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

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

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

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

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

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

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

Monitore o Astra Trident

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

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

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

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

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

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

O que você vai precisar

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

Passo 1: Defina um alvo Prometheus

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

Passo 2: Crie um Prometheus ServiceMonitor

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

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

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

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

Passo 3: Consultar métricas do Trident com PromQL

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

Aqui estão algumas consultas PromQL que você pode usar:

Obtenha informações de saúde do Trident

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

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

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

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

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

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

Obtenha informações de uso do Astra Trident

- **Tamanho médio do volume**

```
trident_volume_allocated_bytes/trident_volume_count
```

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

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

Obtenha uso de volume individual

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

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

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Saiba mais sobre a telemetria do Astra Trident AutoSupport

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

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

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

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


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

Desativar métricas do Astra Trident

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

Astra Trident para Docker

Pré-requisitos para implantação

Você precisa instalar e configurar os pré-requisitos de protocolo necessários no seu host antes de implantar o Astra Trident.

- Verifique se sua implantação atende a todos "requisitos"os .
- Verifique se você tem uma versão suportada do Docker instalada. Se a versão do Docker estiver desatualizada, "instale ou atualize-o.".

```
docker --version
```

- Verifique se os pré-requisitos do protocolo estão instalados e configurados no seu host:

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

Protocolo	Sistema operacional	Comandos

ISCSI	RHEL/CentOS 7	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`

Protocolo	Sistema operacional	Comandos

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` `defaults` em .
		
		5. Certifique-se de que `open-iscsi` e `multipath-tools` estão ativados e em execução:
		
		`sudo systemctl status multipath-tools`

Implante o Astra Trident

O Astra Trident para Docker fornece integração direta com o ecossistema Docker para plataformas de storage da NetApp. Ele dá suporte ao provisionamento e gerenciamento de recursos de storage da plataforma de storage para hosts Docker, com uma estrutura para adicionar plataformas adicionais no futuro.

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

Várias instâncias do Astra Trident podem ser executadas simultaneamente no mesmo host. Isso permite conexões simultâneas a vários sistemas de armazenamento e tipos de armazenamento, com a capacidade de personalizar o armazenamento usado para os volumes Docker.

O que você vai precisar

Consulte "pré-requisitos para implantação". Depois de garantir que os pré-requisitos sejam atendidos, você estará pronto para implantar o Astra Trident.

Método de plug-in gerenciado Docker (versão 1,13/17,03 e posterior)

Antes de começar

Se você usou o Astra Trident pré Docker 1,13/17,03 no método daemon tradicional, certifique-se de parar o processo Astra Trident e reiniciar seu daemon Docker antes de usar o método do plugin gerenciado.

1. Parar todas as instâncias em execução:

```
pkill /usr/local/bin/netappdvp
pkill /usr/local/bin/trident
```

2. Reinicie o Docker.

```
systemctl restart docker
```

3. Certifique-se de que tem o Docker Engine 17,03 (novo 1,13) ou posterior instalado.

```
docker --version
```

Se a sua versão estiver desatualizada, "instale ou atualize a instalação".

Passos

1. Crie um arquivo de configuração e especifique as opções da seguinte forma:
   - `config`: O nome do arquivo padrão é `config.json`, no entanto, você pode usar qualquer nome que você escolher especificando a `config` opção com o nome do arquivo. O arquivo de configuração deve estar localizado `/etc/netappdvp` no diretório no sistema host.
   - `log-level`: Especifique o nível de registo (`debug`, `info`, `warn` `error` , , `fatal`). A predefinição é `info`.
   - `debug`: Especifique se o log de depuração está ativado. O padrão é falso. Substitui o nível de log, se verdadeiro.

i. Crie um local para o arquivo de configuração:

```
sudo mkdir -p /etc/netappdvp
```

ii. Crie o arquivo de configuração:

```
cat << EOF > /etc/netappdvp/config.json
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",
    "aggregate": "aggr1"
}
EOF
```

2. Inicie o Astra Trident usando o sistema de plugins gerenciado.

```
docker plugin install --grant-all-permissions --alias netapp
netapp/trident-plugin:21.07 config=myConfigFile.json
```

3. Comece a usar o Astra Trident para consumir storage do sistema configurado.
   a. Crie um volume chamado "firstvolume":

```
docker volume create -d netapp --name firstVolume
```

   b. Crie um volume padrão quando o contentor for iniciado:

```
docker run --rm -it --volume-driver netapp --volume
secondVolume:/my_vol alpine ash
```

   c. Remover o volume "firstvolume":

```
docker volume rm firstVolume
```

Método tradicional (versão 1,12 ou anterior)

Antes de começar

1. Certifique-se de que você tem o Docker versão 1,10 ou posterior.

   ```
   docker --version
   ```

   Se a sua versão estiver desatualizada, atualize a instalação.

   ```
   curl -fsSL https://get.docker.com/ | sh
   ```

   Ou, "siga as instruções para sua distribuição".

2. Certifique-se de que NFS e/ou iSCSI estão configurados para o seu sistema.

Passos

1. Instale e configure o plug-in de volume do Docker do NetApp:
   a. Baixe e descompacte o aplicativo:

      ```
      wget
      https://github.com/NetApp/trident/releases/download/v21.04.0/trident-
      installer-21.07.0.tar.gz
      tar zxf trident-installer-21.07.0.tar.gz
      ```

   b. Mover para um local no caminho do compartimento:

      ```
      sudo mv trident-installer/extras/bin/trident /usr/local/bin/
      sudo chown root:root /usr/local/bin/trident
      sudo chmod 755 /usr/local/bin/trident
      ```

   c. Crie um local para o arquivo de configuração:

      ```
      sudo mkdir -p /etc/netappdvp
      ```

   d. Crie o arquivo de configuração:

```
cat << EOF > /etc/netappdvp/ontap-nas.json
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",
    "aggregate": "aggr1"
}
EOF
```

2. Depois de colocar o binário e criar o(s) arquivo(s) de configuração, inicie o daemon Trident usando o arquivo de configuração desejado.

```
sudo trident --config=/etc/netappdvp/ontap-nas.json
```

> A menos que especificado, o nome padrão para o driver de volume é "NetApp".

Depois que o daemon é iniciado, você pode criar e gerenciar volumes usando a interface CLI do Docker

3. Criar um volume:

```
docker volume create -d netapp --name trident_1
```

4. Provisione um volume Docker ao iniciar um contentor:

```
docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol
alpine ash
```

5. Remover um volume Docker:

```
docker volume rm trident_1
docker volume rm trident_2
```

Inicie o Astra Trident na inicialização do sistema

Um arquivo de unidade de exemplo para sistemas baseados em systemd pode ser encontrado `contrib/trident.service.example` no repositório Git. Para usar o arquivo com CentOS/RHEL, faça o seguinte:

1. Copie o arquivo para o local correto.

   Você deve usar nomes exclusivos para os arquivos de unidade se tiver mais de uma instância em execução.

   ```
   cp contrib/trident.service.example
   /usr/lib/systemd/system/trident.service
   ```

2. Edite o arquivo, altere a descrição (linha 2) para corresponder ao nome do driver e ao caminho do arquivo de configuração (linha 9) para refletir seu ambiente.
3. Recarregue systemd para que ele ingere alterações:

   ```
   systemctl daemon-reload
   ```

4. Ative o serviço.

   Esse nome varia dependendo do que você nomeou o arquivo no `/usr/lib/systemd/system` diretório.

   ```
   systemctl enable trident
   ```

5. Inicie o serviço.

   ```
   systemctl start trident
   ```

6. Ver o estado.

   ```
   systemctl status trident
   ```

Sempre que você modificar o arquivo unit, execute o `systemctl daemon-reload` comando para que ele esteja ciente das alterações.

Atualize ou desinstale o Astra Trident

Você pode atualizar com segurança o Astra Trident para Docker sem qualquer impacto nos volumes que estão em uso. Durante o processo de atualização, haverá um breve período em que `docker volume` os comandos direcionados para o plugin não serão bem-sucedidos, e os aplicativos não poderão montar volumes até que o plugin esteja sendo executado novamente. Na maioria das circunstâncias, esta é uma questão de segundos.

Atualização

Execute as etapas abaixo para atualizar o Astra Trident para Docker.

Passos

1. Listar os volumes existentes:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

2. Desativar o plugin:

```
docker plugin disable -f netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Atualize o plugin:

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```

> O lançamento de 18,01 do Astra Trident substitui o nDVP. Você deve atualizar diretamente da `netapp/ndvp-plugin` imagem para a `netapp/trident-plugin` imagem.

4. Ativar o plugin:

```
docker plugin enable netapp:latest
```

5. Verifique se o plugin está ativado:

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       Trident - NetApp Docker Volume
Plugin   true
```

6. Verifique se os volumes estão visíveis:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

Se você estiver atualizando de uma versão antiga do Astra Trident (pré-20,10) para Astra Trident 20,10 ou posterior, talvez haja um erro. Para obter mais informações, "Problemas conhecidos"consulte . Se você correr para o erro, você deve primeiro desativar o plugin, em seguida, remover o plugin e, em seguida, instalar a versão necessária do Astra Trident passando um parâmetro de configuração extra: `docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all-permissions config=config.json`

Desinstalar

Execute as etapas abaixo para desinstalar o Astra Trident para Docker.

Passos

1. Remova todos os volumes criados pelo plugin.
2. Desativar o plugin:

```
docker plugin disable netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Remova o plugin:

```
docker plugin rm netapp:latest
```

Trabalhe com volumes

Você pode criar, clonar e remover volumes facilmente usando os comandos padrão `docker volume` com o nome do driver Astra Trident especificado quando necessário.

Crie um volume

- Crie um volume com um driver usando o nome padrão:

```
docker volume create -d netapp --name firstVolume
```

- Criar um volume com uma instância específica do Astra Trident:

```
docker volume create -d ntap_bronze --name bronzeVolume
```

Se você não especificar nenhum "opções", os padrões para o driver serão usados.

- Substituir o tamanho de volume predefinido. Veja o exemplo a seguir para criar um volume 20GiB com um driver:

```
docker volume create -d netapp --name my_vol --opt size=20G
```

Os tamanhos de volume são expressos como strings contendo um valor inteiro com unidades opcionais (exemplo: 10g, 20GB, 3TiB). Se nenhuma unidade for especificada, o padrão é G. as unidades de tamanho podem ser expressas como potências de 2 (B, KiB, MIB, GiB, TIB) ou potências de 10 (B, KB, MB, GB, TB). As unidades shorthand usam poderes de 2 (G GiB, T TIB,…).

Remova um volume

- Remova o volume como qualquer outro volume do Docker:

```
docker volume rm firstVolume
```

Ao utilizar o `solidfire-san` controlador, o exemplo acima elimina e elimina o volume.

Execute as etapas abaixo para atualizar o Astra Trident para Docker.

Clonar um volume

Ao usar o `ontap-nas`, `ontap-san`, , `solidfire-san` e `gcp-cvs storage drivers`, o Astra Trident pode clonar volumes. Ao usar os `ontap-nas-flexgroup` drivers ou `ontap-nas-economy`, a clonagem não é suportada. Criar um novo volume a partir de um volume existente resultará na criação de um novo instantâneo.

- Inspecione o volume para enumerar instantâneos:

```
docker volume inspect <volume_name>
```

- Crie um novo volume a partir de um volume existente. Isso resultará na criação de um novo snapshot:

```
docker volume create -d <driver_name> --name <new_name> -o
from=<source_docker_volume>
```

- Criar um novo volume a partir de um instantâneo existente em um volume. Isso não criará um novo snapshot:

```
docker volume create -d <driver_name> --name <new_name> -o
from=<source_docker_volume> -o fromSnapshot=<source_snap_name>
```

Exemplo

```
[me@host ~]$ docker volume inspect firstVolume

[
    {
        "Driver": "ontap-nas",
        "Labels": null,
        "Mountpoint": "/var/lib/docker-volumes/ontap-
nas/netappdvp_firstVolume",
        "Name": "firstVolume",
        "Options": {},
        "Scope": "global",
        "Status": {
            "Snapshots": [
                {
                    "Created": "2017-02-10T19:05:00Z",
                    "Name": "hourly.2017-02-10_1505"
                }
            ]
        }
    }
]

[me@host ~]$ docker volume create -d ontap-nas --name clonedVolume -o
from=firstVolume
clonedVolume

[me@host ~]$ docker volume rm clonedVolume
[me@host ~]$ docker volume create -d ontap-nas --name volFromSnap -o
from=firstVolume -o fromSnapshot=hourly.2017-02-10_1505
volFromSnap

[me@host ~]$ docker volume rm volFromSnap
```

Acesse volumes criados externamente

Você pode acessar dispositivos de bloco criados externamente (ou seus clones) por contentores usando Trident **only** se eles não tiverem partições e se seu sistema de arquivos for suportado pelo Astra Trident (por exemplo: Um `ext4`-formatado `/dev/sdc1` não será acessível via Astra Trident).

Opções de volume específicas do condutor

Cada driver de armazenamento tem um conjunto diferente de opções, que você pode especificar no momento da criação do volume para personalizar o resultado. Veja abaixo as opções que se aplicam ao sistema de armazenamento configurado.

Usar essas opções durante a operação de criação de volume é simples. Forneça a opção e o valor usando o `-o` operador durante a operação CLI. Estes substituem quaisquer valores equivalentes do arquivo de configuração JSON.

Opções de volume ONTAP

As opções de criação de volume para NFS e iSCSI incluem o seguinte:

Opção	Descrição
`size`	O tamanho do volume, padrão é 1 GiB.
`spaceReserve`	Provisionamento fino ou espesso do volume, o padrão é fino. Os valores válidos são `none` (thin Provisioning) e `volume` (thick provisioned).
`snapshotPolicy`	Isto irá definir a política de instantâneos para o valor pretendido. O padrão é `none`, o que significa que nenhum instantâneo será criado automaticamente para o volume. A menos que seja modificada pelo administrador de storage, existe uma política chamada "padrão" em todos os sistemas ONTAP, que cria e retém seis snapshots por hora, dois por dia e dois por semana. Os dados preservados em um snapshot podem ser recuperados navegando para `.snapshot` o diretório em qualquer diretório do volume.
`snapshotReserve`	Isto irá definir a reserva de instantâneos para a percentagem pretendida. O padrão não é nenhum valor, o que significa que o ONTAP selecionará o snapshotServe (geralmente 5%) se você selecionou uma política de snapshotPolicy, ou 0% se a política de snapshotPolicy não for nenhuma. Você pode definir o valor padrão snapshotServe no arquivo de configuração para todos os backends ONTAP, e você pode usá-lo como uma opção de criação de volume para todos os backends ONTAP, exceto ONTAP-nas-economy.

Opção	Descrição
`splitOnClone`	Ao clonar um volume, isso fará com que o ONTAP divida imediatamente o clone de seu pai. A predefinição é `false`. Alguns casos de uso para clonagem de volumes são melhor servidos dividindo o clone de seu pai imediatamente após a criação, porque é improvável que haja alguma oportunidade de eficiência de storage. Por exemplo, clonar um banco de dados vazio pode oferecer grande economia de tempo, mas pouca economia de armazenamento, por isso é melhor dividir o clone imediatamente.
`encryption`	Isso ativará 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. Atualmente, a criptografia agregada do NetApp (NAE) não é suportada no Trident.
`tieringPolicy`	Define a política de disposição em categorias a ser usada para o volume. Isso decide se os dados são movidos para a categoria de nuvem quando ficam inativos (frios).

As seguintes opções adicionais são para NFS **somente**:

Opção	Descrição
`unixPermissions`	Isso controla o conjunto de permissões para o próprio volume. Por padrão, as permissões serão definidas como `---rwxr-xr-x`, ou em notação numérica 0755, e `root` serão o proprietário. O texto ou o formato numérico funcionarão.
`snapshotDir`	Definir isso como `true` tornará o `.snapshot` diretório visível para os clientes que acessam o volume. O valor padrão é `false`, o que significa que a visibilidade `.snapshot` do diretório está desativada por padrão. Algumas imagens, por exemplo, a imagem oficial do MySQL, não funcionam como esperado quando o `.snapshot` diretório está visível.
`exportPolicy`	Define a política de exportação a ser utilizada para o volume. A predefinição é `default`.

Opção	Descrição
`securityStyle`	Define o estilo de segurança a ser usado para acesso ao volume. A predefinição é `unix`. Os valores válidos são `unix` e `mixed`.

As seguintes opções adicionais são para iSCSI **somente**:

Opção	Descrição
`fileSystemType`	Define o sistema de ficheiros utilizado para formatar volumes iSCSI. A predefinição é `ext4`. Os valores válidos são `ext3`, `ext4`, e `xfs`.
`spaceAllocation`	Definir isso como `false` desativará o recurso de alocação de espaço do LUN. O valor padrão é `true`, o que significa que o ONTAP notifica o host quando o volume ficou sem espaço e o LUN no volume não pode aceitar gravações. Essa opção também permite que o ONTAP recupere espaço automaticamente quando o host exclui dados.

Exemplos

Veja os exemplos abaixo:

- Criar um volume 10GiBD:

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- Criar um volume 100GiBD com instantâneos:

```
docker volume create -d netapp --name demo -o size=100G -o
snapshotPolicy=default -o snapshotReserve=10
```

- Crie um volume que tenha o bit setuid ativado:

```
docker volume create -d netapp --name demo -o unixPermissions=4755
```

O tamanho mínimo do volume é 20MiB.

Se a reserva de snapshot não for especificada e a política de snapshot for `none`, o Trident usará uma reserva de snapshot de 0%.

- Criar um volume sem política de snapshot e sem reserva de snapshot:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- Crie um volume sem política de snapshot e uma reserva de snapshot personalizada de 10%:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- Crie um volume com uma política de snapshot e uma reserva de snapshot personalizada de 10%:

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy --opt snapshotReserve=10
```

- Crie um volume com uma política de snapshot e aceite a reserva de snapshot padrão do ONTAP (geralmente 5%):

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy
```

Opções de volume do software Element

As opções de software Element expõem as políticas de tamanho e qualidade do serviço (QoS) associadas ao volume. Quando o volume é criado, a política de QoS associada a ele é especificada usando a `-o type=service_level` nomenclatura.

A primeira etapa para definir um nível de serviço QoS com o driver Element é criar pelo menos um tipo e especificar o IOPS mínimo, máximo e de pico associado a um nome no arquivo de configuração.

Outras opções de criação de volume de software Element incluem o seguinte:

Opção	Descrição
`size`	O tamanho do volume, padrão para 1GiB ou entrada de configuração … "Padrões": 5G.
`blocksize`	Use 512 ou 4096, o padrão é 512 ou a entrada de configuração DefaultBlockSize.

Exemplo

Veja o seguinte arquivo de configuração de exemplo com definições de QoS:

```
{
    "...": "..."
    "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
            }
        }
    ]
}
```

Na configuração acima, temos três definições de política: Bronze, prata e ouro. Esses nomes são arbitrários.

- Criar um volume 10GiB Gold:

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- Criar um volume Bronze 100GiB:

```
docker volume create -d solidfire --name sfBronze -o type=Bronze -o
size=100G
```

CVS nas opções de volume do GCP

As opções de criação de volume para o CVS no driver do GCP incluem o seguinte:

Opção	Descrição
`size`	O tamanho do volume, por padrão, é 100 GiB para volumes CVS-Performance ou 300 GiB para volumes CVS.
`serviceLevel`	O nível de serviço CVS do volume, por padrão, é padrão. Valores válidos são padrão, premium e extremos.
`snapshotReserve`	Isto irá definir a reserva de instantâneos para a percentagem pretendida. O padrão não é nenhum valor, o que significa que o CVS selecionará a reserva de snapshot (geralmente 0%).

Exemplos

- Criar um volume 2TiBD:

```
docker volume create -d netapp --name demo -o size=2T
```

- Crie um volume premium de 5TiB TB:

```
docker volume create -d netapp --name demo -o size=5T -o
serviceLevel=premium
```

O tamanho mínimo de volume é de 100 GiB para volumes CVS-Performance ou de 300 GiB para volumes CVS.

Opções de volume Azure NetApp Files

As opções de criação de volume para o driver Azure NetApp Files incluem o seguinte:

Opção	Descrição
`size`	O tamanho do volume, por padrão, é de 100 GB.

Exemplos

- Criar um volume 200GiBD:

```
docker volume create -d netapp --name demo -o size=200G
```

O tamanho mínimo do volume é de 100 GB.

Recolher registos

Você pode coletar Registros para obter ajuda com a solução de problemas. O método que você usa para coletar os logs varia de acordo com a forma como você está executando o plugin Docker.

Recolha registos para resolução de problemas

Passos

1. Se você estiver executando o Astra Trident usando o método de plug-in gerenciado recomendado (ou seja, usando `docker plugin` comandos), visualize-os da seguinte forma:

```
# docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
4fb97d2b956b        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
# journalctl -u docker | grep 4fb97d2b956b
```

   O nível de registo padrão deve permitir diagnosticar a maioria dos problemas. Se você achar que isso não é suficiente, você pode ativar o Registro de depuração.

2. Para ativar o registo de depuração, instale o plug-in com o registo de depuração ativado:

```
docker plugin install netapp/trident-plugin:<version> --alias <alias>
debug=true
```

   Ou, ative o registo de depuração quando o plug-in já estiver instalado:

```
docker plugin disable <plugin>
docker plugin set <plugin> debug=true
docker plugin enable <plugin>
```

3. Se você estiver executando o binário em si no host, os logs estarão disponíveis no diretório do host `/var/log/netappdvp`. Para ativar o registo de depuração, especifique `-debug` quando executar o plugin.

Dicas gerais de solução de problemas

- O problema mais comum em que novos usuários são executados é uma configuração incorreta que impede que o plugin seja inicializado. Quando isso acontecer, você provavelmente verá uma mensagem como esta quando você tentar instalar ou ativar o plugin:

  ```
  Error response from daemon: dial unix /run/docker/plugins/<id>/netapp.sock:
  connect: no such file or directory
  ```

Isto significa que o plugin falhou ao iniciar. Felizmente, o plugin foi construído com uma capacidade de Registro abrangente que deve ajudá-lo a diagnosticar a maioria dos problemas que você provavelmente encontrará.

- Se houver problemas com a montagem de um PV em um recipiente, certifique-se de que `rpcbind` está instalado e funcionando. Use o gerenciador de pacotes necessário para o sistema operacional do host e verifique se `rpcbind` está em execução. Você pode verificar o status do serviço rpcbind executando um `systemctl status rpcbind` ou seu equivalente.

Gerenciar várias instâncias do Astra Trident

Várias instâncias do Trident são necessárias quando você deseja ter várias configurações de storage disponíveis simultaneamente. A chave para várias instâncias é dar nomes diferentes usando a `--alias` opção com o plug-in em contentor, ou `--volume-driver` opção ao instanciar o Trident no host.

Etapas para o plugin gerenciado do Docker (versão 1,13/17,03 ou posterior)

1. Inicie a primeira instância especificando um alias e um arquivo de configuração.

```
docker plugin install --grant-all-permissions --alias silver
netapp/trident-plugin:21.07 config=silver.json
```

2. Inicie a segunda instância, especificando um alias diferente e arquivo de configuração.

```
docker plugin install --grant-all-permissions --alias gold
netapp/trident-plugin:21.07 config=gold.json
```

3. Crie volumes especificando o alias como o nome do driver.

   Por exemplo, para o volume de ouro:

```
docker volume create -d gold --name ntapGold
```

   Por exemplo, para o volume prateado:

```
docker volume create -d silver --name ntapSilver
```

Passos para o tradicional (versão 1,12 ou anterior)

1. Inicie o plugin com uma configuração NFS usando um ID de driver personalizado:

```
sudo trident --volume-driver=netapp-nas --config=/path/to/config
-nfs.json
```

2. Inicie o plug-in com uma configuração iSCSI usando um ID de driver personalizado:

```
sudo trident --volume-driver=netapp-san --config=/path/to/config
-iscsi.json
```

3. Provisione volumes Docker para cada instância de driver:

   Por exemplo, para NFS:

```
docker volume create -d netapp-nas --name my_nfs_vol
```

   Por exemplo, para iSCSI:

```
docker volume create -d netapp-san --name my_iscsi_vol
```

Opções de configuração de armazenamento

Consulte as opções de configuração disponíveis para suas configurações do Astra Trident.

Opções de configuração global

Essas opções de configuração se aplicam a todas as configurações do Astra Trident, independentemente da plataforma de storage usada.

Opção	Descrição	Exemplo
`version`	Número da versão do ficheiro de configuração	1
`storageDriverName`	Nome do driver de armazenamento	`ontap-nas` `ontap-san`, , `ontap-nas-economy` `ontap-nas-flexgroup` , `solidfire-san`, , , `azure-netapp-files`, ou `gcp-cvs`
`storagePrefix`	Prefixo opcional para nomes de volume. Padrão: "Netappdvp_".	estadiamento_
`limitVolumeSize`	Restrição opcional nos tamanhos de volume. Padrão: "" (não aplicado)	13 10 g

Não use `storagePrefix` (incluindo o padrão) para backends de elemento. Por padrão, o `solidfire-san` driver ignorará essa configuração e não usará um prefixo. Recomendamos usar um tenantID específico para mapeamento de volume do Docker ou usar os dados de atributo que são preenchidos com a versão do Docker, informações de driver e nome bruto do Docker nos casos em que qualquer nome munging pode ter sido usado.

As opções padrão estão disponíveis para evitar ter que especificá-las em cada volume criado. A `size` opção está disponível para todos os tipos de controlador. Consulte a seção Configuração do ONTAP para obter um exemplo de como definir o tamanho padrão do volume.

Opção	Descrição	Exemplo
`size`	Tamanho padrão opcional para novos volumes. Padrão: "1G"	10G

Configuração ONTAP

Além dos valores de configuração global acima, ao usar o ONTAP, as seguintes opções de nível superior estão disponíveis.

Opção	Descrição	Exemplo
`managementLIF`	Endereço IP do ONTAP Management LIF. Você pode especificar um nome de domínio totalmente qualificado (FQDN).	10.0.0.1
`dataLIF`	Endereço IP do protocolo LIF; será derivado se não for especificado. Para os `ontap-nas` drivers **somente**, você pode especificar um FQDN, caso em que o FQDN será usado para as operações de montagem NFS. Para os `ontap-san` drivers, o padrão é usar todos os IPs de LIF de dados da SVM e usar multipath iSCSI. Especificar um endereço IP para `dataLIF` os `ontap-san` drivers força o driver a desabilitar o multipath e usar apenas o endereço especificado.	10.0.0.2
`svm`	Máquina virtual de armazenamento a utilizar (necessária, se o LIF de gestão for um LIF de cluster)	svm_nfs
`username`	Nome de utilizador para ligar ao dispositivo de armazenamento	vsadmin

Opção	Descrição	Exemplo
`password`	Palavra-passe para ligar ao dispositivo de armazenamento	segredo
`aggregate`	Agregado para provisionamento (opcional; se definido, deve ser atribuído ao SVM). Para `ontap-nas-flexgroup` o driver, essa opção é ignorada. Todos os agregados atribuídos ao SVM são usados para provisionar um volume FlexGroup.	aggr1
`limitAggregateUsage`	Opcional, falha no provisionamento se o uso estiver acima dessa porcentagem	75%
`nfsMountOptions`	Controle refinado das opções de montagem NFS; o padrão é "-o nfsvers 3". **Disponível apenas para os `ontap-nas` condutores e `ontap-nas-economy`.** "Consulte as informações de configuração do host NFS aqui".	-o nfsvers 4
`igroupName`	O igroup usado pelo plugin; o padrão é "netappdvp". **Disponível apenas para o rio ONTAP-san**.	miigrupo
`limitVolumeSize`	Tamanho máximo do volume requestable e tamanho do volume pai de qtree. **Para o `ontap-nas-economy` driver, essa opção limita adicionalmente o tamanho dos FlexVols que ele cria**.	13 300 g
`qtreesPerFlexvol`	Qtrees máximos por FlexVol, tem de estar no intervalo [50, 300], o padrão é 200. **Para `ontap-nas-economy` o driver, esta opção permite personalizar o número máximo de qtrees por FlexVol**.	300

As opções padrão estão disponíveis para evitar ter que especificá-las em cada volume criado:

Opção	Descrição	Exemplo
`spaceReserve`	Modo de reserva de espaço; "nenhum" (thin Provisioning) ou "volume" (thick)	nenhum
`snapshotPolicy`	A política do Snapshot a ser usada, o padrão é "nenhum"	nenhum
`snapshotReserve`	O padrão é "" para aceitar o padrão do ONTAP	10
`splitOnClone`	Dividir um clone de seu pai na criação, o padrão é "falso"	falso
`encryption`	Ativar encriptação de volume NetApp, por predefinição, "false"	verdadeiro
`unixPermissions`	Opção nas para volumes NFS provisionados, o padrão é "777"	777
`snapshotDir`	Opção nas para acesso ao `.snapshot` diretório, o padrão é "false"	verdadeiro
`exportPolicy`	A opção nas para a política de exportação NFS usar, o padrão é "padrão"	padrão
`securityStyle`	Opção nas para acesso ao volume NFS provisionado, o padrão é "UNIX"	misto
`fileSystemType`	Opção SAN para selecionar o tipo de sistema de arquivos, o padrão é "ext4"	xfs
`tieringPolicy`	A política de disposição em categorias a ser usada, o padrão é "nenhuma"; "somente snapshot" para configuração pré-ONTAP 9.5 SVM-DR	nenhum

Opções de dimensionamento

Os `ontap-nas` drivers e `ontap-san` criam um ONTAP FlexVol para cada volume do Docker. O ONTAP dá suporte a até 1000 FlexVols por nó de cluster com um máximo de cluster de 12.000 FlexVols. Se os requisitos de volume do Docker se ajustarem a essa limitação, `ontap-nas` o driver será a solução nas preferida devido aos recursos adicionais oferecidos pelo FlexVols, como snapshots Docker volume granular e clonagem.

Se você precisar de mais volumes do Docker do que pode ser acomodado pelos limites do FlexVol, escolha o `ontap-nas-economy` ou o `ontap-san-economy` driver.

`ontap-nas-economy`O driver cria volumes do Docker como Qtrees do ONTAP em um pool de FlexVols gerenciados automaticamente. As Qtrees oferecem dimensionamento muito maior, até 100.000 PB por nó de cluster e 2.400.000 PB por cluster, à custa de alguns recursos. `ontap-nas-economy`O driver não oferece suporte a snapshots ou clonagem granular de volume do Docker.

No momento, o `ontap-nas-economy` driver não é compatível com o Docker Swarm, porque o Swarm não orquestra a criação de volume em vários nós.

`ontap-san-economy`O driver cria volumes do Docker como LUNs ONTAP em um pool compartilhado de FlexVols gerenciados automaticamente. Dessa forma, cada FlexVol não se restringe a apenas um LUN e oferece melhor escalabilidade para workloads SAN. Dependendo do storage array, o ONTAP oferece suporte para até 16384 LUNs por cluster. Como os volumes são LUNs abaixo, esse driver oferece suporte a snapshots e clonagem granular do Docker volume.

Escolha o `ontap-nas-flexgroup` driver para aumentar o paralelismo para um único volume que pode crescer para o intervalo de petabytes com bilhões de arquivos. Alguns casos de uso ideais para FlexGroups incluem IA/ML/DL, big data e análise, compilações de software, streaming, repositórios de arquivos e assim por diante. O Trident usa todos os agregados atribuídos a uma SVM ao provisionar um volume FlexGroup. O suporte do FlexGroup no Trident também tem as seguintes considerações:

- Requer ONTAP versão 9,2 ou superior.
- A partir desta redação, FlexGroups só suportam NFS v3.
- Recomendado para ativar os identificadores NFSv3 de 64 bits para o SVM.
- O tamanho mínimo recomendado de FlexGroup é 100GB.
- A clonagem não é compatível com volumes FlexGroup.

Para obter informações sobre FlexGroups e cargas de trabalho apropriadas para FlexGroups, consulte "Guia de práticas recomendadas e implementação de volumes do NetApp FlexGroup".

Para obter recursos avançados e grande escala no mesmo ambiente, você pode executar várias instâncias do Docker volume Plugin, com uma usando `ontap-nas` e outra usando `ontap-nas-economy`o .

Exemplo de arquivos de configuração do ONTAP

Exemplo de NFS para `ontap-nas` driver

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",
    "aggregate": "aggr1",
    "defaults": {
      "size": "10G",
      "spaceReserve": "none",
      "exportPolicy": "default"
    }
}
```

Exemplo de NFS para `ontap-nas-flexgroup` driver

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",
    "defaults": {
      "size": "100G",
      "spaceReserve": "none",
      "exportPolicy": "default"
    }
}
```

Exemplo de NFS para `ontap-nas-economy` driver

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",
    "aggregate": "aggr1"
}
```

Exemplo iSCSI para `ontap-san` driver

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "username": "vsadmin",
    "password": "secret",
    "aggregate": "aggr1",
    "igroupName": "myigroup"
}
```

Exemplo de NFS para `ontap-san-economy` driver

```
{
    "version": 1,
    "storageDriverName": "ontap-san-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi_eco",
    "username": "vsadmin",
    "password": "secret",
    "aggregate": "aggr1",
    "igroupName": "myigroup"
}
```

Configuração do software Element

Além dos valores de configuração global, ao usar o software Element (NetApp HCI/SolidFire), essas opções estão disponíveis.

Opção	Descrição	Exemplo
`Endpoint`	https://<login>:<password>@<mvip>/json-rpc/<element-version>	https://admin:admin@192.168.160.3/json-rpc/8.0
`SVIP`	Endereço IP iSCSI e porta	10,0.0,7:3260
`TenantName`	Locatário do SolidFireF para usar (criado se não for encontrado)	"docker"
`InitiatorIFace`	Especifique a interface ao restringir o tráfego iSCSI a uma interface não predefinida	"padrão"
`Types`	Especificações de QoS	Veja o exemplo abaixo
`LegacyNamePrefix`	Prefixo para instalações Trident atualizadas. Se você usou uma versão do Trident anterior a 1.3.2 e fez uma atualização com volumes existentes, precisará definir esse valor para acessar seus volumes antigos que foram mapeados pelo método de nome de volume.	"netappdvp-"

O `solidfire-san` driver não suporta Docker Swarm.

Exemplo de arquivo de configuração de software Element

```
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://admin:admin@192.168.160.3/json-rpc/8.0",
    "SVIP": "10.0.0.7:3260",
    "TenantName": "docker",
    "InitiatorIFace": "default",
    "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
            }
        }
    ]
}
```

Cloud Volumes Service (CVS) na configuração do GCP

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

Inscreva-se para aceder a volumes inferiores a 1TiB "aqui" .

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

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

- PVCs menores que 300 GiB resultarão na criação de um volume CVS de 300 GiB Trident.
- Os PVCs que estão entre 300 GiB e 600 GiB resultarão na criação de um volume CVS do tamanho solicitado pelo Trident.
- Os PVCs que estão entre 600 GiB e 1 TIB resultarão na criação de um volume CVS de 1TiB TB no Trident.
- PVCs maiores que 1 TIB resultarão na criação de um volume CVS do tamanho solicitado pelo Trident.

Além dos valores de configuração global, ao usar o CVS no GCP, essas opções estão disponíveis.

Opção	Descrição	Exemplo
`apiRegion`	Região da conta CVS (obrigatório). É a região do GCP onde esse back-end provisionará volumes.	"us-west2"
`projectNumber`	Número do projeto GCP (obrigatório). Pode ser encontrado na tela inicial do portal da web do GCP.	"123456789012"
`hostProjectNumber`	Número do projeto do host VPC compartilhado do GCP (necessário se estiver usando uma VPC compartilhada)	"098765432109"
`apiKey`	Chave de API para conta de serviço do GCP com função de administrador do CVS (obrigatório). É o conteúdo formatado em JSON do arquivo de chave privada de uma conta de serviço do GCP (copiado literalmente no arquivo de configuração do back-end). A conta de serviço deve ter a função netappcloudvolumes.admin.	(conteúdo do arquivo de chave privada)
`secretKey`	Chave secreta da conta CVS (obrigatório). Pode ser encontrado no portal da Web CVS em Definições de conta > Acesso à API.	"padrão"

Opção	Descrição	Exemplo
`proxyURL`	URL do proxy se o servidor proxy precisar se conetar à conta CVS. O servidor proxy pode ser um proxy HTTP ou um proxy HTTPS. No caso de um proxy HTTPS, a validação do certificado é ignorada para permitir o uso de certificados autoassinados no servidor proxy. **Servidores proxy com autenticação ativada não são suportados**.	"http://proxy-server-hostname/"
`nfsMountOptions`	Opções de montagem NFS; o padrão é "-o nfsvers 3"	"3,proto tcp,timeo-600"
`serviceLevel`	Nível de desempenho (padrão, premium, Extreme), padrão para "padrão"	"premium"
`network`	A rede do GCP usada para volumes CVS, o padrão é "padrão"	"padrão"

Se estiver usando uma rede VPC compartilhada, você deverá especificar ambos `projectNumber` e `hostProjectNumber`. Nesse caso, `projectNumber` é o projeto de serviço e `hostProjectNumber` é o projeto host.

O NetApp Cloud Volumes Service para GCP não é compatível com volumes CVS-Performance inferiores a 100 GiB de tamanho, nem com volumes CVS inferiores a 300 GiB de tamanho. Para facilitar a implantação de aplicações, o Trident cria automaticamente volumes do tamanho mínimo se for solicitado um volume muito pequeno.

Ao usar o CVS no GCP, essas configurações padrão de opção de volume estão disponíveis.

Opção	Descrição	Exemplo
`exportRule`	Lista de acesso NFS (endereços e/ou sub-redes CIDR), o padrão é "0,0.0,0/0"	"10,0.1,0/24,10.0.2,100"
`snapshotDir`	Controla a visibilidade `.snapshot` do diretório	"falso"
`snapshotReserve`	O padrão é "" para aceitar o padrão CVS de 0	"10"
`size`	Tamanho do volume, padrão para "100GiB"	"10T"

Exemplo CVS no arquivo de configuração do GCP

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

Configuração do Azure NetApp Files

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

- `subscriptionID` A partir de uma subscrição do Azure com o Azure NetApp Files ativado
- `tenantID`, `clientID` E `clientSecret` de um "Registo da aplicação" no Azure ative Directory com permissões suficientes para o serviço Azure NetApp Files
- Localização do Azure que contém pelo menos um "sub-rede delegada"

Se você estiver usando o Azure NetApp Files pela primeira vez ou em um novo local, será necessária alguma configuração inicial para que o o "guia quickstart" o possa guiar.

O Astra Trident 21.04.0 e versões anteriores não são compatíveis com pools de capacidade de QoS manual.

Opção	Descrição	Padrão
`version`	Sempre 1	
`storageDriverName`	"azure-NetApp-files"	
`backendName`	Nome personalizado para o back-end de armazenamento	Nome do condutor e carateres aleatórios
`subscriptionID`	O ID da assinatura da sua assinatura do Azure	
`tenantID`	O ID do locatário de um Registro de aplicativo	
`clientID`	A ID do cliente de um registo de aplicação	
`clientSecret`	O segredo do cliente de um Registro de aplicativo	
`serviceLevel`	Um dos "Standard", "Premium" ou "Ultra"	"" (aleatório)
`location`	O nome do local do Azure novos volumes serão criados no	"" (aleatório)
`virtualNetwork`	Nome de uma rede virtual com uma sub-rede delegada	"" (aleatório)
`subnet`	Nome de uma sub-rede delegada `Microsoft.Netapp/volumes`	"" (aleatório)
`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)

O serviço Azure NetApp Files não oferece suporte a volumes com menos de 100 GB de tamanho. Para facilitar a implantação de aplicativos, o Trident cria automaticamente volumes de 100 GB se um volume menor for solicitado.

Você pode controlar como cada volume é provisionado por padrão usando essas opções em uma seção especial da configuração.

Opção	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`	Controla a visibilidade `.snapshot` do diretório	"falso"
`size`	O tamanho padrão dos novos volumes	"100G"

Exemplo de configurações do Azure NetApp Files

- Exemplo 1: Configuração mínima de back-end para azure-NetApp-Files*

Esta é a configuração mínima absoluta de back-end. Com essa configuração, o Trident descobrirá todas as suas contas do NetApp, pools de capacidade e sub-redes delegadas no ANF em todos os locais do mundo e colocará novos volumes em um deles aleatoriamente.

Essa configuração é útil quando você começar a usar o ANF e experimentar as coisas, mas na prática você vai querer fornecer um escopo adicional para os volumes provisionados para garantir que eles tenham as características desejadas e acabem em em uma rede próxima da computação que está usando. Veja os exemplos subsequentes para obter mais detalhes.

```
{
    "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"
}
```

Exemplo 2: Local único e nível de serviço específico para arquivos azure-NetApp

Essa configuração de back-end coloca volumes no local "eastus" do Azure em um pool de capacidade "Premium". O Trident descobre automaticamente todas as sub-redes delegadas ao ANF nesse local e colocará 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": "Premium"
}
```

Exemplo 3: Configuração avançada para arquivos azure-NetApp

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": "Premium",
    "virtualNetwork": "my-virtual-network",
    "subnet": "my-subnet",
    "nfsMountOptions": "nfsvers=3,proto=tcp,timeo=600",
    "limitVolumeSize": "500Gi",
    "defaults": {
        "exportRule": "10.0.0.0/24,10.0.1.0/24,10.0.2.100",
        "size": "200Gi"
    }
}
```

Exemplo 4: Pools de armazenamento virtual com arquivos azure-NetApp

Essa configuração de back-end define vários "piscinas de armazenamento" 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.

Isso está apenas arranhando a superfície do poder dos pools de armazenamento virtual e suas etiquetas.

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
    "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
    "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
    "clientSecret": "SECRET",
    "nfsMountOptions": "nfsvers=3,proto=tcp,timeo=600",
    "labels": {
        "cloud": "azure"
    },
    "location": "eastus",

    "storage": [
        {
            "labels": {
                "performance": "gold"
            },
            "serviceLevel": "Ultra"
        },
        {
            "labels": {
                "performance": "silver"
            },
            "serviceLevel": "Premium"
        },
        {
            "labels": {
                "performance": "bronze"
            },
            "serviceLevel": "Standard",
        }
    ]
}
```

Problemas e limitações conhecidos

Encontre informações sobre problemas e limitações conhecidos ao usar o Astra Trident com Docker.

A atualização do plug-in de volume do Docker do Trident para 20,10 e posterior a partir de versões mais antigas resulta em falha de atualização com o erro de nenhum arquivo ou diretório.

Solução alternativa

1. Desative o plugin.

```
docker plugin disable -f netapp:latest
```

2. Remova o plugin.

```
docker plugin rm -f netapp:latest
```

3. Reinstale o plugin fornecendo o parâmetro extra `config`.

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant
-all-permissions config=config.json
```

Os nomes dos volumes devem ter um mínimo de 2 carateres.

Esta é uma limitação de cliente Docker. O cliente interpretará um único nome de caractere como sendo um caminho do Windows. "Veja o bug 25773".

O Docker Swarm tem certos comportamentos que impedem o Astra Trident de dar suporte a ele em cada combinação de storage e driver.

- Docker Swarm atualmente faz uso do nome do volume em vez de ID do volume como seu identificador de volume exclusivo.
- As solicitações de volume são enviadas simultaneamente para cada nó em um cluster Swarm.
- Os plug-ins de volume (incluindo o Astra Trident) devem ser executados de forma independente em cada nó em um cluster do Swarm. Devido à forma como o ONTAP funciona e como os `ontap-nas` drivers e `ontap-san` funcionam, eles são os únicos que podem operar dentro dessas limitações.

O resto dos pilotos estão sujeitos a problemas como condições de corrida que podem resultar na criação de um grande número de volumes para uma única solicitação sem um claro "vencedor"; por exemplo, o elemento tem um recurso que permite que os volumes tenham o mesmo nome, mas IDs diferentes.

O NetApp forneceu feedback à equipe do Docker, mas não tem qualquer indicação de recurso futuro.

Se um FlexGroup estiver sendo provisionado, o ONTAP não provisiona um segundo FlexGroup se o segundo FlexGroup tiver um ou mais agregados em comum com o FlexGroup sendo provisionado.

Perguntas frequentes

Encontre respostas para as perguntas mais frequentes sobre a instalação, configuração, atualização e solução de problemas do Astra Trident.

Questões gerais

Com que frequência o Astra Trident é lançado?

O Astra Trident é lançado a cada três meses: Janeiro, abril, julho e outubro. Este é um mês após o lançamento do Kubernetes.

O Astra Trident é compatível com todos os recursos lançados em uma versão específica do Kubernetes?

O Astra Trident geralmente não é compatível com recursos alfa do Kubernetes. O Trident pode oferecer suporte a recursos beta nas duas versões do Trident que seguem a versão beta do Kubernetes.

O Astra Trident tem alguma dependência de outros produtos NetApp para seu funcionamento?

O Astra Trident não tem dependências em outros produtos de software NetApp e funciona como uma aplicação autônoma. No entanto, você deve ter um dispositivo de storage de back-end do NetApp.

Como posso obter detalhes completos da configuração do Astra Trident?

Use o `tridentctl get` comando para obter mais informações sobre sua configuração do Astra Trident.

Posso obter métricas sobre como o storage é provisionado pelo Astra Trident?

Sim. O Trident 20,01 apresenta endpoints Prometheus que podem ser usados para coletar informações sobre a operação do Astra Trident, como o número de backends gerenciados, o número de volumes provisionados, bytes consumidos, etc. Você também pode usar o Cloud Insights para monitoramento e análise.

A experiência do usuário muda ao usar o Astra Trident como um supervisor CSI?

Não há alterações no que diz respeito à experiência do usuário e às funcionalidades. O nome do provisionador usado é `csi.trident.netapp.io`. Esse método de instalação do Astra Trident é recomendado se você quiser usar todos os novos recursos fornecidos pelas versões atuais e futuras.

Instalar e usar o Astra Trident em um cluster Kubernetes

Quais são as versões suportadas `etcd` do ?

O Astra Trident não precisa mais de um `etcd`. ele usa CRDs para manter o estado.

O Astra Trident oferece suporte a uma instalação off-line a partir de um Registro privado?

Sim, o Astra Trident pode ser instalado offline. "aqui"Consulte .

Posso instalar o Astra Trident be remotamente?

Sim. O Astra Trident 18,10 e posterior são compatíveis com a funcionalidade de instalação remota de qualquer máquina que tenha `kubectl` acesso ao cluster. Depois `kubectl` que o acesso for verificado (por exemplo, inicie um `kubectl get nodes` comando da máquina remota para verificar), siga as instruções de instalação.

Posso configurar a alta disponibilidade com o Astra Trident?

O Astra Trident é instalado como uma implantação do Kubernetes (ReplicaSet) com uma instância, e por isso tem HA incorporada. Você não deve aumentar o número de réplicas na implantação. Se o nó em que o Astra Trident está instalado for perdido ou o pod estiver inacessível, o Kubernetes reimplanta automaticamente o pod em um nó íntegro no cluster. O Astra Trident é apenas um plano de controle. Portanto, os pods montados atualmente não são afetados se o Astra Trident for reimplantado.

O Astra Trident precisa de acesso ao namespace do sistema kube?

O Astra Trident lê o servidor de API Kubernetes para determinar quando as aplicações solicitam novos PVCs de modo que a TI precisa de acesso ao sistema kube.

Quais são as funções e Privileges usadas pelo Astra Trident?

O instalador do Trident cria um ClusterRole Kubernetes, que tem acesso específico aos recursos Persistentvolume, PersistentVolumeClaim, StorageClass e segredo do cluster do Kubernetes. "aqui"Consulte .

Posso gerar localmente os arquivos de manifesto exatos que o Astra Trident usa para instalação?

Você pode gerar e modificar localmente os arquivos de manifesto exatos que o Astra Trident usa para instalação, se necessário. "aqui"Consulte .

Posso compartilhar o mesmo SVM de back-end do ONTAP em duas instâncias separadas do Astra Trident em dois clusters Kubernetes separados?

Embora não seja aconselhado, você pode usar o mesmo SVM de back-end em duas instâncias do Astra Trident. Especifique um nome de volume exclusivo para cada instância durante a instalação e/ou especifique um parâmetro exclusivo `StoragePrefix` no `setup/backend.json` arquivo. Isso serve para garantir que o mesmo FlexVol não seja usado para ambas as instâncias.

É possível instalar o Astra Trident sob o ContainerLinux (antigo CoreOS)?

O Astra Trident é simplesmente um pod de Kubernetes e pode ser instalado onde quer que o Kubernetes esteja em execução.

Posso usar o Astra Trident com NetApp Cloud Volumes ONTAP?

Sim, o Astra Trident é compatível com AWS, Google Cloud e Azure.

O Astra Trident funciona com o Cloud volumes Services?

Sim, o Astra Trident é compatível com o serviço Azure NetApp Files no Azure e com o Cloud Volumes Service no GCP.

Solução de problemas e suporte

O NetApp é compatível com Astra Trident?

Embora o Astra Trident seja de código aberto e fornecido gratuitamente, o NetApp oferece suporte total desde que o back-end do NetApp seja compatível.

Como faço para levantar um caso de suporte?

Para levantar um caso de suporte, execute um dos seguintes procedimentos:

1. Entre em Contato com seu gerente de conta de suporte e obtenha ajuda para levantar um ticket.
2. Envie um caso de suporte entrando em Contato "Suporte à NetApp"com .

Como gerar um pacote de log de suporte?

Você pode criar um pacote de suporte executando `tridentctl logs -a`o . Além dos logs capturados no pacote, capture o log do kubelet para diagnosticar os problemas de montagem no lado do Kubernetes. As instruções para obter o log do kubelet variam de acordo com a forma como o Kubernetes é instalado.

O que devo fazer se for necessário enviar uma solicitação para um novo recurso?

Crie um problema "Trident GitHub" e mencione **RFE** no assunto e na descrição do problema.

Onde posso levantar um defeito?

Crie um problema no "Astra Trident GitHub". Certifique-se de incluir todas as informações e logs necessários relativos ao problema.

O que acontece se eu tiver uma pergunta rápida sobre o Astra Trident sobre a qual preciso de esclarecimentos? Existe uma comunidade ou um fórum?

Se você tiver dúvidas, problemas ou solicitações, entre em Contato conosco através de nossa "Folga" equipe ou GitHub.

A senha do meu sistema de storage mudou e o Astra Trident não funciona mais. Como faço para recuperar?

Atualize a senha do backend com `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`o . Substitua `myBackend` no exemplo pelo nome do backend e `/path/to_new_backend.json` pelo caminho para o arquivo correto `backend.json`.

O Astra Trident não encontra meu nó Kubernetes. Como faço para corrigir isso?

Há dois cenários prováveis pelos quais o Astra Trident não consegue encontrar um nó Kubernetes. Pode ser devido a um problema de rede no Kubernetes ou a um problema de DNS. O daemonset do nó do Trident que

é executado em cada nó do Kubernetes deve ser capaz de se comunicar com o controlador Trident para Registrar o nó no Trident. Se as alterações de rede ocorrerem após a instalação do Astra Trident, você encontrará esse problema apenas com novos nós Kubernetes adicionados ao cluster.

Se o pod Trident for destruído, eu perderei os dados?

Os dados não serão perdidos se o pod Trident for destruído. Os metadados do Trident são armazenados em objetos CRD. Todos os PVS que foram provisionados pelo Trident funcionarão normalmente.

Atualizar o Astra Trident

Posso atualizar de uma versão mais antiga diretamente para uma versão mais recente (ignorando algumas versões)?

A NetApp oferece suporte à atualização do Astra Trident de um grande lançamento para o próximo grande lançamento imediato. Você pode atualizar da versão 18.xx para 19.xx, 19.xx para 20.xx, e assim por diante. Você deve testar a atualização em um laboratório antes da implantação da produção.

É possível fazer o downgrade do Trident para uma versão anterior?

Há uma série de fatores a serem avaliados se você quiser fazer downgrade. "a seção sobre downgrade"Consulte .

Gerenciar backends e volumes

Preciso definir o gerenciamento e LIFs de dados em um arquivo de definição de back-end do ONTAP?

O NetApp recomenda ter ambos no arquivo de definição de back-end. No entanto, o Management LIF é o único que é obrigatório.

O Astra Trident pode configurar o CHAP para backends ONTAP?

Sim. A partir de 20,04, o Astra Trident suporta CHAP bidirecional para backends ONTAP. Isso requer configuração `useCHAP=true` em sua configuração de back-end.

Como faço para gerenciar políticas de exportação com o Astra Trident?

O Astra Trident pode criar e gerenciar políticas de exportação dinamicamente a partir da versão 20,04. Isso permite que o administrador de storage forneça um ou mais blocos CIDR em sua configuração de back-end e que o Trident adicione IPs de nós que se enquadram nesses intervalos a uma política de exportação criada por ele. Dessa forma, o Astra Trident gerencia automaticamente a adição e exclusão de regras para nós com IPs nos CIDR fornecidos. Este recurso requer o CSI Trident.

Podemos especificar uma porta no DataLIF?

O Astra Trident 19,01 e posterior suportam a especificação de uma porta no DataLIF. Configure-o `backend.json` no arquivo como `"managementLIF": <ip address>:<port>"`. Por exemplo, se o endereço IP do LIF de gerenciamento for 192,0.2,1 e a porta for 1000, configure `"managementLIF": "192.0.2.1:1000"`o .

Os endereços IPv6 podem ser usados para os LIFs de gerenciamento e dados?

Sim. O Astra Trident 20,01 suporta a definição de endereços IPv6 para os parâmetros de gerenciamento de LIF e dataLIF para backends ONTAP. Você deve garantir que o endereço segue a semântica IPv6 e que o managementLIF é definido entre colchetes (por exemplo, `[ec0d:6504:a9c1:ae67:53d1:4bdf:ab32:e233]`). Você também deve garantir que o Astra Trident esteja instalado usando o `--use-ipv6` sinalizador para que ele funcione em mais de IPv6.

É possível atualizar o LIF de gerenciamento no back-end?

Sim, é possível atualizar o backend Management LIF usando o `tridentctl update backend` comando.

É possível atualizar o Data LIF no backend?

Não, não é possível atualizar o Data LIF no backend.

Posso criar vários back-ends no Astra Trident para Kubernetes?

O Astra Trident pode dar suporte a muitos backends simultaneamente, seja com o mesmo driver ou com drivers diferentes.

Como o Astra Trident armazena credenciais de back-end?

O Astra Trident armazena as credenciais de back-end como segredos do Kubernetes.

Como o Astra Trident seleciona um back-end específico?

Se os atributos de back-end não puderem ser usados para selecionar automaticamente os pools corretos para uma classe, os `storagePools` parâmetros e `additionalStoragePools` serão usados para selecionar um conjunto específico de pools.

Como posso garantir que o Astra Trident não provisione de um back-end específico?

O `excludeStoragePools` parâmetro é usado para filtrar o conjunto de pools que o Astra Trident usará para provisionar e removerá todos os pools correspondentes.

Se houver vários backends do mesmo tipo, como o Astra Trident seleciona qual back-end usar?

Se houver vários backends configurados do mesmo tipo, o Astra Trident seleciona o back-end apropriado com base nos parâmetros presentes no `StorageClass` e `PersistentVolumeClaim`no . Por exemplo, se houver vários backends de driver ONTAP-nas, o Astra Trident tentará corresponder parâmetros no `StorageClass` e `PersistentVolumeClaim` combinou e corresponder a um back-end que possa atender aos requisitos listados em `StorageClass` e `PersistentVolumeClaim`. Se houver vários backends que correspondam à solicitação, o Astra Trident seleciona um deles aleatoriamente.

O Astra Trident é compatível com CHAP bidirecional com Element/SolidFire?

Sim.

Como o Astra Trident implanta Qtrees em um volume ONTAP? Quantos Qtrees podem ser implantados em um único volume?

`ontap-nas-economy`O driver cria até 200 Qtrees no mesmo FlexVol (configurável entre 50 e 300), 100.000 Qtrees por nó de cluster e 2,4M por cluster. Quando você insere um novo `PersistentVolumeClaim` que é atendido pelo driver de economia, o driver procura ver se já existe um FlexVol que pode atender o novo Qtree. Se o FlexVol não existir que possa servir o Qtree, um novo FlexVol será criado.

Como posso definir permissões Unix para volumes provisionados no ONTAP nas?

Você pode definir permissões Unix no volume provisionado pelo Astra Trident definindo um parâmetro no arquivo de definição de back-end.

Como posso configurar um conjunto explícito de opções de montagem ONTAP NFS enquanto provisiono um volume?

Por padrão, o Astra Trident não define as opções de montagem como nenhum valor com o Kubernetes. Para especificar as opções de montagem na classe de armazenamento do Kubernetes, siga o exemplo fornecido "aqui".

Como faço para definir os volumes provisionados para uma política de exportação específica?

Para permitir que os hosts apropriados acessem um volume, use o `exportPolicy` parâmetro configurado no arquivo de definição de back-end.

Como definir a criptografia de volumes por meio do Astra Trident com ONTAP?

Você pode definir a criptografia no volume provisionado pelo Trident usando o parâmetro de criptografia no arquivo de definição de back-end.

Qual é a melhor maneira de implementar QoS para ONTAP por meio do Astra Trident?

`StorageClasses`Use para implementar QoS para ONTAP.

Como especificar o provisionamento thin ou thick por meio do Astra Trident?

Os drivers ONTAP oferecem suporte ao provisionamento thin ou thick. Os drivers do ONTAP são padrão para thin Provisioning. Se o provisionamento espesso for desejado, você deverá configurar o arquivo de definição de back-end ou o `StorageClass`. Se ambos estiverem configurados, `StorageClass` tem precedência. Configure o seguinte para o ONTAP:

1. On `StorageClass`, defina o `provisioningType` atributo como thick (espesso).
2. No arquivo de definição de back-end, ative volumes espessos definindo `backend spaceReserve`

`parameter` como volume.

Como posso garantir que os volumes que estão a ser utilizados não sejam eliminados mesmo que elimine acidentalmente o PVC?

A proteção de PVC é ativada automaticamente no Kubernetes a partir da versão 1,10.

Posso expandir PVCs de NFS criados pelo Astra Trident?

Sim. Você pode expandir um PVC que foi criado pelo Astra Trident. Observe que o volume com crescimento automático é um recurso do ONTAP que não é aplicável ao Trident.

Se eu tiver um volume criado fora do Astra Trident, posso importá-lo para o Astra Trident?

A partir de 19,04, você pode usar o recurso de importação de volume para levar volumes para o Kubernetes.

Posso importar um volume enquanto estiver no modo de proteção de dados (DP) da SnapMirror ou offline?

A importação de volume falha se o volume externo estiver no modo DP ou estiver offline. Você recebe a seguinte mensagem de erro:

```
Error: could not import volume: volume import failed to get size of
volume: volume <name> was not found (400 Bad Request) command terminated
with exit code 1.
Make sure to remove the DP mode or put the volume online before importing
the volume.
```

Posso expandir PVCs iSCSI criados pelo Astra Trident?

O Trident 19,10 suporta a expansão de PVS iSCSI usando o Supervisor de CSI.

Como a cota de recursos é traduzida para um cluster NetApp?

A cota de recursos de armazenamento do Kubernetes deve funcionar enquanto o armazenamento do NetApp tiver capacidade. Quando o storage do NetApp não consegue atender às configurações de cota do Kubernetes devido à falta de capacidade, o Astra Trident tenta provisionar, mas faz erros.

Posso criar snapshots de volume usando o Astra Trident?

Sim. A criação de snapshots de volume sob demanda e volumes persistentes a partir de snapshots é compatível com o Astra Trident. Para criar PVS a partir de instantâneos, certifique-se de que a `VolumeSnapshotDataSource` porta de recurso foi ativada.

Quais são os drivers compatíveis com snapshots de volume Astra Trident?

A partir de hoje, o suporte a snapshot sob demanda está disponível para o nosso `ontap-nas` `ontap-nas-flexgroup` , `ontap-san`, `ontap-san-economy`, , , `solidfire-san` `gcp-cvs`, e `azure-netapp-files` drivers de back-end.

Como faço para fazer um backup instantâneo de um volume provisionado pelo Astra Trident com ONTAP?

Isso está disponível nos `ontap-nas` drivers , `ontap-san` e `ontap-nas-flexgroup` . Você também pode especificar um `snapshotPolicy` para o `ontap-san-economy` driver no nível FlexVol.

Isso também está disponível `ontap-nas-economy` nos drivers, mas na granularidade de nível FlexVol e não na granularidade de nível de qtree. Para habilitar a capacidade de snapshot volumes provisionados pelo Astra Trident, defina a opção de parâmetro de back-end `snapshotPolicy` para a política de snapshot desejada, conforme definido no back-end do ONTAP. Todos os snapshots feitos pelo controlador de storage não são conhecidos pelo Astra Trident.

Posso definir uma porcentagem de reserva de snapshot para um volume provisionado por meio do Astra Trident?

Sim, você pode reservar uma porcentagem específica de espaço em disco para armazenar as cópias snapshot por meio do Astra Trident definindo `snapshotReserve` o atributo no arquivo de definição de back-end. Se você configurou `snapshotPolicy` e `snapshotReserve` no arquivo de definição de back-end, a porcentagem de reserva de snapshot é definida de acordo com a `snapshotReserve` porcentagem mencionada no arquivo de back-end. Se o `snapshotReserve` número percentual não for mencionado, ONTAP por padrão leva a porcentagem de reserva de snapshot como 5. Se a `snapshotPolicy` opção estiver definida como None (nenhum), a percentagem de reserva de instantâneos é definida como 0.

Posso acessar diretamente o diretório instantâneo do volume e copiar arquivos?

Sim, você pode acessar o diretório instantâneo no volume provisionado pelo Trident definindo o `snapshotDir` parâmetro no arquivo de definição de back-end.

Posso configurar o SnapMirror para volumes com o Astra Trident?

Atualmente, o SnapMirror precisa ser definido externamente usando a CLI ou o OnCommand System Manager do ONTAP.

Como faço para restaurar volumes persistentes para um snapshot específico do ONTAP?

Para restaurar um volume para um instantâneo do ONTAP, execute as seguintes etapas:

1. Quiesce o pod do aplicativo que está usando o volume persistente.
2. Reverter para o snapshot necessário por meio da CLI ou OnCommand System Manager do ONTAP.
3. Reinicie o pod de aplicativos.

O Trident provisiona volumes em SVMs que têm um espelhamento de compartilhamento de carga configurado?

Os espelhos de compartilhamento de carga podem ser criados para volumes raiz de SVMs que fornecem dados por NFS. O ONTAP atualiza automaticamente os espelhos de compartilhamento de carga para volumes criados pelo Trident. Isso pode resultar em atrasos nos volumes de montagem. Quando vários volumes são criados usando o Trident, o provisionamento de um volume depende da atualização do espelhamento de compartilhamento de carga do ONTAP.

Como posso separar o uso da classe de storage para cada cliente/locatário?

O Kubernetes não permite classes de storage em namespaces. No entanto, você pode usar o Kubernetes para limitar o uso de uma classe de armazenamento específica por namespace usando cotas de recursos de armazenamento, que são por namespace. Para negar acesso a um namespace específico a um armazenamento específico, defina a cota de recurso como 0 para essa classe de armazenamento.

Suporte

O Astra Trident é um projeto NetApp com suporte oficial. Você pode entrar em Contato com o NetApp usando qualquer um dos mecanismos padrão e obter o suporte de nível empresarial de que você precisa.

Há também uma vibrante comunidade pública de usuários de contêineres (incluindo desenvolvedores do Astra Trident) no `containers` canal no "O trabalho do NetApp no Slack". Este é um ótimo lugar para fazer perguntas gerais sobre o projeto e discutir tópicos relacionados com colegas de mentalidade semelhante.

Solução de problemas

Use os ponteiros fornecidos aqui para solucionar problemas que você pode encontrar ao instalar e usar o Astra Trident.

Para obter ajuda com o Astra Trident, crie um pacote de suporte usando `tridentctl logs -a -n trident` e envie para `NetApp Support <Getting Help>`o .

Para obter uma lista abrangente de artigos de solução de problemas, consulte "Base de Conhecimento NetApp (login necessário)" . Você também encontrará informações sobre a solução de problemas relacionados ao Astra "aqui".

Resolução de problemas gerais

- Se o pod Trident não aparecer corretamente (por exemplo, quando o pod Trident está preso `ContainerCreating` na fase com menos de dois contêineres prontos), em execução `kubectl -n trident describe deployment trident` e `kubectl -n trident describe pod trident-**` pode fornecer informações adicionais. A obtenção de logs do kubelet (por exemplo, via `journalctl -xeu kubelet`) também pode ser útil.
- Se não houver informações suficientes nos logs do Trident, você pode tentar ativar o modo de depuração para o Trident passando o `-d` sinalizador para o parâmetro de instalação com base na opção de instalação.

  Em seguida, confirme se a depuração é definida usando `./tridentctl logs -n trident` e pesquisando `level=debug msg` no log.

  Instalado com Operador

  ```
  # kubectl patch torc trident -n <namespace> --type=merge -p
  '{"spec":{"debug":true}}'
  ```

  Isso reiniciará todos os pods do Trident, o que pode levar vários segundos. Você pode verificar isso observando a coluna 'IDADE' na saída do `kubectl get pod -n trident`.

  Para Astra Trident 20,07 e 20,10 utilização `tprov` em vez de `torc` .

  Instalado com Helm

  ```
  $ helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
  tridentDebug=true`
  ```

  Instalado com tridentctl

  ```
  ./tridentctl uninstall -n trident
  ./tridentctl install -d -n trident
  ```

- Você também pode obter logs de depuração para cada back-end, incluindo `debugTraceFlags` na sua

definição de back-end. Por exemplo, inclua `debugTraceFlags: {"api":true, "method":true,}` para obter chamadas de API e transversais de método nos logs do Trident. Os backends existentes podem ter `debugTraceFlags` sido configurados com um `tridentctl backend update`.

- Ao usar o RedHat CoreOS, certifique-se de que `iscsid` está habilitado nos nós de trabalho e iniciado por padrão. Isso pode ser feito usando OpenShift MachineConfigs ou modificando os modelos de ignição.
- Um problema comum que você pode encontrar ao usar o Trident com "Azure NetApp Files" é quando os segredos do locatário e do cliente vêm de um Registro de aplicativo com permissões insuficientes. Para obter uma lista completa dos requisitos do Trident, consulte "Azure NetApp Files" Configuração.
- Se houver problemas com a montagem de um PV em um recipiente, certifique-se de que `rpcbind` está instalado e funcionando. Use o gerenciador de pacotes necessário para o sistema operacional do host e verifique se `rpcbind` está em execução. Você pode verificar o status `rpcbind` do serviço executando um `systemctl status rpcbind` ou seu equivalente.
- Se um back-end do Trident relatar que ele está `failed` no estado apesar de ter trabalhado antes, provavelmente será causado pela alteração das credenciais do SVM/administrador associadas ao back-end. Atualizar as informações de back-end usando `tridentctl update backend` ou saltando o pod Trident corrigirá esse problema.
- Se você estiver atualizando seu cluster Kubernetes e/ou Trident para usar snapshots de volume beta, certifique-se de que todos os CRS de snapshot alfa existentes sejam completamente removidos. Em seguida, você pode usar o `tridentctl obliviate alpha-snapshot-crd` comando para excluir CRDs de snapshot alfa. "este blog"Consulte para compreender as etapas envolvidas na migração de snapshots alfa.
- Se você encontrar problemas de permissão ao instalar o Trident com Docker como o runtime do contentor, tente a instalação do Trident com o `--in cluster=false` sinalizador. Isso não usará um pod do instalador e evitará problemas de permissão vistos devido ao `trident-installer` usuário.
- Utilize o `uninstall parameter <Uninstalling Trident>` para limpar após uma falha de funcionamento. Por padrão, o script não remove os CRDs que foram criados pelo Trident, tornando seguro desinstalar e instalar novamente mesmo em uma implantação em execução.
- Se você estiver procurando fazer o downgrade para uma versão anterior do Trident, primeiro execute o `tridenctl uninstall` comando para remover o Trident. Baixe o desejado "Versão Trident" e instale usando o `tridentctl install` comando. Considere apenas um downgrade se não houver novos PVS criados e se não houver alterações em PVs/backends/classes de armazenamento já existentes. Como o Trident agora usa CRDs para manter o estado, todas as entidades de armazenamento criadas (backends, classes de armazenamento, PVS e instantâneos de volume) têm `associated CRD objects <Kubernetes CustomResourceDefinition Objects>` em vez de dados gravados no PV que foram usados pela versão instalada anterior do Trident. **PVS recém-criados não serão utilizáveis ao voltar para uma versão anterior. Alterações feitas em objetos, como backends, PVS, classes de armazenamento e snapshots de volume (criados/atualizados/excluídos) não serão visíveis para o Trident quando desclassificados**. O PV que foi usado pela versão anterior do Trident instalado ainda será visível para o Trident. Voltar para uma versão anterior não interromperá o acesso para PVS que já foram criados usando a versão mais antiga, a menos que tenham sido atualizados.
- Para remover completamente o Trident, execute o `tridentctl obliviate crd` comando. Isso removerá todos os objetos CRD e desdefinirá as CRDs. O Trident não gerenciará mais nenhum PVS que já tenha provisionado.

  Depois disso, o Trident precisará ser reconfigurado do zero.

- Após uma instalação bem-sucedida, se um PVC estiver preso na `Pending` fase, a execução `kubectl describe pvc` pode fornecer informações adicionais sobre por que a Trident não conseguiu fornecer um PV para este PVC.

Solução de problemas de uma implantação de Trident mal sucedida usando o operador

Se você estiver implantando o Trident usando o operador, o status das `TridentOrchestrator` alterações será alterado de `Installing` para `Installed`. Se você observar o `Failed` status e o operador não conseguir se recuperar sozinho, você deve verificar os logs do operador executando o seguinte comando:

```
tridentctl logs -l trident-operator
```

Arrastar os logs do contentor do operador Trident pode apontar para onde está o problema. Por exemplo, um desses problemas poderia ser a incapacidade de extrair as imagens de contentor necessárias de Registros upstream em um ambiente com airgapped.

Para entender por que a instalação do Trident não foi bem-sucedida, você deve dar uma olhada no `TridentOrchestrator` status.

```
$ kubectl describe torc trident-2
Name:         trident-2
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Status:
  Current Installation Params:
    IPv6:
    Autosupport Hostname:
    Autosupport Image:
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:
    Enable Node Prep:
    Image Pull Secrets:         <nil>
    Image Registry:
    k8sTimeout:
    Kubelet Dir:
    Log Format:
    Silence Autosupport:
    Trident Image:
  Message:                      Trident is bound to another CR 'trident'
  Namespace:                    trident-2
  Status:                       Error
  Version:
Events:
  Type     Reason  Age                From                          Message
  ----     ------  ----               ----                          -------
  Warning  Error   16s (x2 over 16s)  trident-operator.netapp.io  Trident
is bound to another CR 'trident'
```

Este erro indica que já existe um `TridentOrchestrator` que foi usado para instalar o Trident. Como cada cluster do Kubernetes pode ter apenas uma instância do Trident, o operador garante que, a qualquer momento, só exista uma ativa `TridentOrchestrator` que possa criar.

Além disso, observar o status dos pods do Trident geralmente pode indicar se algo não está certo.

```
$ kubectl get pods -n trident

NAME                                READY   STATUS             RESTARTS
AGE
trident-csi-4p5kq                   1/2     ImagePullBackOff   0
5m18s
trident-csi-6f45bfd8b6-vfrkw        4/5     ImagePullBackOff   0
5m19s
trident-csi-9q5xc                   1/2     ImagePullBackOff   0
5m18s
trident-csi-9v95z                   1/2     ImagePullBackOff   0
5m18s
trident-operator-766f7b8658-ldzsv   1/1     Running            0
8m17s
```

Você pode ver claramente que os pods não são capazes de inicializar completamente porque uma ou mais imagens de contentor não foram obtidas.

Para resolver o problema, você deve editar o `TridentOrchestrator` CR. Alternativamente, você pode excluir `TridentOrchestrator` e criar um novo com a definição modificada e precisa.

Solução de problemas de uma implantação do Trident mal sucedida usando `tridentctl`

Para ajudar a descobrir o que deu errado, você pode executar o instalador novamente usando o `-d` argumento, que irá ativar o modo de depuração e ajudá-lo a entender qual é o problema:

```
./tridentctl install -n trident -d
```

Depois de resolver o problema, você pode limpar a instalação da seguinte forma e, em seguida, executar o `tridentctl install` comando novamente:

```
./tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Removed Trident user from security context constraint.
INFO Trident uninstallation succeeded.
```

Práticas recomendadas e recomendações

Implantação

Use as recomendações listadas aqui quando implantar o Astra Trident.

Implante em um namespace dedicado

"Namespaces" fornecer separação administrativa entre diferentes aplicações e são uma barreira para o compartilhamento de recursos. Por exemplo, um PVC de um namespace não pode ser consumido de outro. O Astra Trident fornece recursos PV para todos os namespaces no cluster do Kubernetes e, consequentemente, utiliza uma conta de serviço que elevou o Privileges.

Além disso, o acesso ao pod Trident pode permitir que um usuário acesse credenciais do sistema de storage e outras informações confidenciais. É importante garantir que os usuários de aplicativos e aplicativos de gerenciamento não tenham a capacidade de acessar as definições de objetos do Trident ou os próprios pods.

Use cotas e limites de intervalo para controlar o consumo de armazenamento

O Kubernetes tem dois recursos que, quando combinados, fornecem um mecanismo avançado para limitar o consumo de recursos pelas aplicações. O "mecanismo de cota de storage" permite que o administrador implemente limites de consumo globais e específicos de classe de storage, de contagem de objetos e capacidade em uma base por namespace. Além disso, o uso de a "limite de alcance" garante que as solicitações de PVC estejam dentro de um valor mínimo e máximo antes que a solicitação seja encaminhada para o provisionador.

Esses valores são definidos em uma base por namespace, o que significa que cada namespace deve ter valores definidos que se encaixam em seus requisitos de recursos. Consulte aqui para obter informações "como alavancar cotas" sobre .

Configuração de armazenamento

Cada plataforma de storage do portfólio do NetApp tem funcionalidades exclusivas que beneficiam aplicações, em contêineres ou não. O Trident funciona com ONTAP e Element. Não há uma plataforma que seja mais adequada para todos os aplicativos e cenários do que outra, no entanto, as necessidades do aplicativo e da equipe que administra o dispositivo devem ser levadas em conta ao escolher uma plataforma.

Você deve seguir as práticas recomendadas de linha de base para o sistema operacional host com o protocolo que você está utilizando. Opcionalmente, você pode considerar a incorporação de práticas recomendadas de aplicativos, quando disponíveis, com configurações de backend, classe de armazenamento e PVC para otimizar o armazenamento para aplicativos específicos.

Práticas recomendadas de ONTAP e Cloud Volumes ONTAP

Conheça as práticas recomendadas para configurar o ONTAP e o Cloud Volumes ONTAP for Trident.

As recomendações a seguir são diretrizes para configuração do ONTAP para workloads em contêineres, que consomem volumes provisionados dinamicamente pelo Trident. Cada um deve ser considerado e avaliado quanto à adequação em seu ambiente.

Use SVM(s) dedicados ao Trident

As máquinas virtuais de storage (SVMs) fornecem isolamento e separação administrativa entre locatários em um sistema ONTAP. A dedicação de um SVM a aplicações permite a delegação do Privileges e permite aplicar práticas recomendadas para limitar o consumo de recursos.

Há várias opções disponíveis para o gerenciamento do SVM:

- Fornecer a interface de gerenciamento de cluster na configuração de back-end, juntamente com as credenciais apropriadas, e especificar o nome da SVM.
- Crie uma interface de gerenciamento dedicada ao SVM com o Gerenciador de sistemas do ONTAP ou a CLI.
- Compartilhe a função de gerenciamento com uma interface de dados NFS.

Em cada caso, a interface deve estar em DNS, e o nome DNS deve ser usado ao configurar o Trident. Isso ajuda a facilitar alguns cenários de DR, por exemplo, SVM-DR sem o uso de retenção de identidade de rede.

No entanto, não há preferência entre ter um LIF de gerenciamento dedicado ou compartilhado para o SVM, você deve garantir que suas políticas de segurança de rede estejam alinhadas com a abordagem escolhida. Independentemente disso, o LIF de gerenciamento deve ser acessível via DNS para facilitar a máxima flexibilidade deve "SVM-DR" ser usado em conjunto com o Trident.

Limite a contagem máxima de volume

Os sistemas de storage ONTAP têm uma contagem de volume máxima, que varia de acordo com a versão do software e a plataforma de hardware. "NetApp Hardware Universe"Consulte para obter a sua plataforma específica e a versão do ONTAP para determinar os limites exatos. Quando a contagem de volume está esgotada, as operações de provisionamento falham não apenas para o Trident, mas para todas as solicitações de storage.

Os Trident `ontap-nas` e `ontap-san` os drivers provisionam um Flexvolume para cada volume persistente (PV) do Kubernetes criado. O `ontap-nas-economy` driver cria aproximadamente um Flexvolume para cada 200 PVS (configurável entre 50 e 300). O `ontap-san-economy` driver cria aproximadamente um Flexvolume para cada 100 PVS (configurável entre 50 e 200). Para evitar que o Trident consuma todos os volumes disponíveis no sistema de storage, defina um limite para o SVM. Você pode fazer isso a partir da linha de comando:

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

O valor para `max-volumes` varia com base em vários critérios específicos para o seu ambiente:

- O número de volumes existentes no cluster do ONTAP
- O número de volumes que você espera provisionar fora do Trident para outras aplicações
- O número de volumes persistentes esperado para ser consumido pelas aplicações Kubernetes

O `max-volumes` valor é o total de volumes provisionados em todos os nós do cluster ONTAP e não em um nó ONTAP individual. Como resultado, você pode encontrar algumas condições em que um nó de cluster do ONTAP pode ter muito mais ou menos volumes provisionados pelo Trident do que outro nó.

Por exemplo, um cluster ONTAP de dois nós tem a capacidade de hospedar um máximo de 2000 volumes flexíveis. Ter a contagem de volume máxima definida para 1250 parece muito razoável. No entanto, se apenas

"agregados" de um nó forem atribuídos à SVM, ou se os agregados atribuídos de um nó não puderem ser provisionados (por exemplo, devido à capacidade), o outro nó se tornará o destino de todos os volumes provisionados pelo Trident. Isso significa que o limite de volume pode ser alcançado para esse nó antes que o `max-volumes` valor seja atingido, resultando em impacto no Trident e em outras operações de volume que usam esse nó. **Você pode evitar essa situação garantindo que os agregados de cada nó no cluster sejam atribuídos ao SVM usado pelo Trident em números iguais.**

Limite o tamanho máximo de volumes criados pelo Trident

Para configurar o tamanho máximo para volumes que podem ser criados pelo Trident, use o `limitVolumeSize` parâmetro em `backend.json` sua definição.

Além de controlar o tamanho do volume no storage array, você também deve utilizar os recursos do Kubernetes.

Configure o Trident para usar o CHAP bidirecional

Você pode especificar o iniciador CHAP e os nomes de usuário e senhas de destino em sua definição de back-end e ter o Trident Enable CHAP no SVM. Usando o `useCHAP` parâmetro em sua configuração de back-end, o Trident autentica conexões iSCSI para backends ONTAP com CHAP. O suporte CHAP bidirecional está disponível com o Trident 20,04 e superior.

Criar e usar uma política de QoS SVM

A utilização de uma política de QoS ONTAP aplicada à SVM limita o número de consumíveis de IOPS pelos volumes provisionados pelo Trident. Isso ajuda "evite um bully" a evitar que o volume fora de controle afete workloads fora do SVM do Trident.

Você pode criar uma política de QoS para o SVM em algumas etapas. Consulte a documentação da sua versão do ONTAP para obter as informações mais precisas. O exemplo abaixo cria uma política de QoS que limita o total de IOPS disponível para o SVM a 5000.

```
# create the policy group for the SVM
qos policy-group create -policy-group <policy_name> -vserver <svm_name>
-max-throughput 5000iops

# assign the policy group to the SVM, note this will not work
# if volumes or files in the SVM have existing QoS policies
vserver modify -vserver <svm_name> -qos-policy-group <policy_name>
```

Além disso, se a sua versão do ONTAP for compatível com ela, considere o uso de um mínimo de QoS para garantir uma taxa de transferência para workloads em contêineres. A QoS adaptável não é compatível com uma política de nível SVM.

O número de IOPS dedicado aos workloads em contêineres depende de muitos aspectos. Entre outras coisas, estas incluem:

- Outros workloads que usam o storage array. Se houver outras cargas de trabalho, não relacionadas à implantação do Kubernetes, utilizando os recursos de storage, deve-se tomar cuidado para garantir que essas cargas de trabalho não sejam acidentalmente afetadas.
- Workloads esperados em execução em contêineres. Se os workloads com requisitos de IOPS altos forem

executados em contêineres, uma política de QoS baixa resulta em uma experiência ruim.

É importante lembrar que uma política de QoS atribuída no nível SVM resulta em todos os volumes provisionados ao SVM que compartilham o mesmo pool de IOPS. Se uma, ou um número pequeno, das aplicações em contêiner tiverem um requisito de IOPS alto, isso pode se tornar um bully para os outros workloads em contêiner. Se esse for o caso, você pode considerar o uso de automação externa para atribuir políticas de QoS por volume.

> Você deve atribuir o grupo de políticas de QoS ao SVM **somente** se a versão do ONTAP for anterior a 9,8.

Criar grupos de política de QoS para Trident

A qualidade do serviço (QoS) garante que a performance de workloads essenciais não é degradada pelos workloads da concorrência. Os grupos de política de QoS do ONTAP fornecem opções de QoS para volumes e permitem que os usuários definam o limite máximo de taxa de transferência para um ou mais workloads. Para obter mais informações sobre QoS, "Garantir taxa de transferência com QoS" consulte . É possível especificar grupos de políticas de QoS no back-end ou em um pool de storage, e eles são aplicados a cada volume criado nesse pool ou back-end.

O ONTAP tem dois tipos de grupos de política de QoS: Tradicional e adaptável. Os grupos de políticas tradicionais fornecem uma taxa de transferência máxima fixa (ou mínima, em versões posteriores) em IOPS. O serviço adaptável dimensiona automaticamente a taxa de transferência para o tamanho do workload, mantendo a taxa de IOPS para TBs|GBs conforme o tamanho do workload muda. Isso proporciona uma vantagem significativa ao gerenciar centenas ou milhares de workloads em uma implantação grande.

Considere o seguinte ao criar grupos de política de QoS:

- Você deve definir a `qosPolicy` chave no `defaults` bloco da configuração de back-end. Veja o seguinte exemplo de configuração de back-end:

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "0.0.0.0",
  "dataLIF": "0.0.0.0",
  "svm": "svm0",
  "username": "user",
  "password": "pass",
  "defaults": {
    "qosPolicy": "standard-pg"
  },
  "storage": [
    {
      "labels": {"performance": "extreme"},
      "defaults": {
        "adaptiveQosPolicy": "extremely-adaptive-pg"
      }
    },
    {
      "labels": {"performance": "premium"},
      "defaults": {
        "qosPolicy": "premium-pg"
      }
    }
  ]
}
```

- Você deve aplicar os grupos de políticas por volume, para que cada volume obtenha toda a taxa de transferência, conforme especificado pelo grupo de políticas. Grupos de políticas compartilhadas não são suportados.

Para obter mais informações sobre grupos de políticas de QoS, "Comandos de QoS ONTAP 9.8" consulte .

Limitar o acesso a recursos de storage aos membros do cluster do Kubernetes

Limitar o acesso aos volumes NFS e iSCSI LUNs criados pelo Trident é um componente essencial da postura de segurança para a implantação do Kubernetes. Isso impede que os hosts que não fazem parte do cluster do Kubernetes acessem os volumes e potencialmente modifiquem os dados inesperadamente.

É importante entender que os namespaces são o limite lógico dos recursos no Kubernetes. A suposição é que os recursos no mesmo namespace são capazes de ser compartilhados, no entanto, é importante, não há capacidade entre namespace. Isso significa que, embora os PVS sejam objetos globais, quando vinculados a um PVC, eles só são acessíveis por pods que estão no mesmo namespace. **É fundamental garantir que os namespaces sejam usados para fornecer separação quando apropriado.**

A principal preocupação da maioria das organizações com relação à segurança de dados em um contexto do Kubernetes é que um processo em um contêiner pode acessar o storage montado no host, mas que não se destina ao contêiner. "Namespaces" foram concebidos para evitar este tipo de compromisso. No entanto, há

uma exceção: Contentores privilegiados.

Um contentor privilegiado é aquele que é executado com permissões substancialmente mais no nível do host do que o normal. Estes não são negados por padrão, portanto, certifique-se de desativar a capacidade "diretivas de segurança do pod" usando o .

Para volumes em que o acesso é desejado tanto do Kubernetes quanto de hosts externos, o storage deve ser gerenciado de maneira tradicional, com o PV introduzido pelo administrador e não gerenciado pelo Trident. Isso garante que o volume de storage seja destruído somente quando o Kubernetes e os hosts externos forem desconetados e não estiverem mais usando o volume. Além disso, é possível aplicar uma política de exportação personalizada, que permite o acesso dos nós de cluster do Kubernetes e dos servidores direcionados fora do cluster do Kubernetes.

Para implantações que têm nós de infraestrutura dedicados (por exemplo, OpenShift) ou outros nós que não são agendáveis para aplicativos de usuário, políticas de exportação separadas devem ser usadas para limitar ainda mais o acesso aos recursos de armazenamento. Isso inclui a criação de uma política de exportação para serviços que são implantados nesses nós de infraestrutura (por exemplo, os serviços de métricas e Registro OpenShift) e aplicativos padrão que são implantados em nós que não são de infraestrutura.

Use uma política de exportação dedicada

Você deve garantir que existe uma política de exportação para cada back-end que permita somente o acesso aos nós presentes no cluster do Kubernetes. O Trident pode criar e gerenciar automaticamente políticas de exportação a partir da versão 20,04. Dessa forma, o Trident limita o acesso aos volumes provisionados por TI aos nós no cluster do Kubernetes e simplifica a adição/exclusão de nós.

Como alternativa, você também pode criar uma política de exportação manualmente e preenchê-la com uma ou mais regras de exportação que processam cada solicitação de acesso de nó:

- Use o `vserver export-policy create` comando ONTAP CLI para criar a política de exportação.
- Adicione regras à política de exportação usando o `vserver export-policy rule create` comando ONTAP CLI.

Executar esses comandos permite restringir quais nós do Kubernetes têm acesso aos dados.

`showmount`Desativar o SVM da aplicação

O `showmount` recurso permite que um cliente NFS consulte o SVM para obter uma lista de exportações de NFS disponíveis. Um pod implantado no cluster do Kubernetes pode emitir o `showmount -e` comando contra o LIF de dados e receber uma lista de montagens disponíveis, incluindo aquelas às quais ele não tem acesso. Embora isso, por si só, não seja um compromisso de segurança, ele fornece informações desnecessárias potencialmente ajudando um usuário não autorizado a se conetar a uma exportação NFS.

Você deve desativar `showmount` usando o comando ONTAP CLI no nível da SVM:

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

Práticas recomendadas da SolidFire

Conheça as práticas recomendadas para configurar o armazenamento SolidFire para Trident.

Crie uma conta SolidFire

Cada conta do SolidFire representa um proprietário de volume exclusivo e recebe seu próprio conjunto de credenciais do Protocolo de Autenticação de desafio-aperto (CHAP). Você pode acessar volumes atribuídos a uma conta usando o nome da conta e as credenciais CHAP relativas ou por meio de um grupo de acesso de volume. Uma conta pode ter até dois mil volumes atribuídos a ela, mas um volume pode pertencer a apenas uma conta.

Crie uma política de QoS

Use as políticas de qualidade do serviço (QoS) do SolidFire se quiser criar e salvar uma configuração padronizada de qualidade do serviço que pode ser aplicada a muitos volumes.

Você pode definir parâmetros de QoS em uma base por volume. O desempenho de cada volume pode ser garantido definindo três parâmetros configuráveis que definem a QoS: Min IOPS, Max IOPS e Burst IOPS.

Aqui estão os possíveis valores de IOPS mínimo, máximo e de pico sazonal para o tamanho de bloco 4Kb.

Parâmetro IOPS	Definição	Valor mín	Valor padrão	Valor máximo (4Kb)
IOPS mín	O nível garantido de desempenho para um volume.	50	50	15000
IOPS máx	O desempenho não excederá este limite.	50	15000	200.000
IOPS de explosão	Máximo de IOPS permitido em um cenário de pico curto.	50	15000	200.000

Embora o IOPS máximo e o IOPS Burst possam ser definidos até 200.000 K, o desempenho máximo real de um volume é limitado pelo uso do cluster e pelo desempenho por nó.

O tamanho do bloco e a largura de banda têm uma influência direta no número de IOPS. À medida que os tamanhos de blocos aumentam, o sistema aumenta a largura de banda para um nível necessário para processar os tamanhos de blocos maiores. À medida que a largura de banda aumenta, o número de IOPS que o sistema consegue atingir diminui. Consulte "SolidFire qualidade do serviço" para obter mais informações sobre QoS e desempenho.

Autenticação SolidFire

O Element suporta dois métodos de autenticação: CHAP e volume Access Groups (VAG). O CHAP usa o protocolo CHAP para autenticar o host no back-end. Os grupos de acesso de volume controlam o acesso aos volumes que ele provisiona. O NetApp recomenda usar o CHAP para autenticação, pois é mais simples e não tem limites de escala.

O Trident com o provisionador de CSI aprimorado suporta o uso da autenticação CHAP. Os VAG só devem ser utilizados no modo de funcionamento tradicional não CSI.

A autenticação CHAP (verificação de que o iniciador é o usuário de volume pretendido) é suportada apenas

com controle de acesso baseado em conta. Se você estiver usando CHAP para autenticação, duas opções estão disponíveis: CHAP unidirecional e CHAP bidirecional. O CHAP unidirecional autentica o acesso ao volume usando o nome da conta do SolidFire e o segredo do iniciador. A opção CHAP bidirecional fornece a maneira mais segura de autenticar o volume porque o volume autentica o host através do nome da conta e do segredo do iniciador e, em seguida, o host autentica o volume através do nome da conta e do segredo de destino.

No entanto, se o CHAP não puder ser ativado e os VAG forem necessários, crie o grupo de acesso e adicione os iniciadores e volumes do host ao grupo de acesso. Cada IQN que você adicionar a um grupo de acesso pode acessar cada volume no grupo com ou sem autenticação CHAP. Se o iniciador iSCSI estiver configurado para usar autenticação CHAP, o controle de acesso baseado em conta será usado. Se o iniciador iSCSI não estiver configurado para usar a autenticação CHAP, o controle de acesso ao grupo de acesso de volume será usado.

Onde encontrar mais informações?

Alguns dos documentos de melhores práticas estão listados abaixo. PESQUISE na "Biblioteca NetApp" para as versões mais atuais.

ONTAP

- "Guia de práticas recomendadas e implementação de NFS"
- "Guia de administração DE SAN" (Para iSCSI)
- "Configuração iSCSI Express para RHEL"

Software Element

- "Configurando o SolidFire para Linux"

NetApp HCI

- "Pré-requisitos de implantação do NetApp HCI"
- "Acesse o mecanismo de implantação do NetApp"

Informações sobre as melhores práticas de aplicação

- "Melhores práticas para MySQL no ONTAP"
- "Melhores práticas para MySQL no SolidFire"
- "NetApp SolidFire e Cassandra"
- "Práticas recomendadas da Oracle no SolidFire"
- "Melhores práticas do PostgreSQL no SolidFire"

Nem todos os aplicativos têm diretrizes específicas, é importante trabalhar com sua equipe do NetApp e usar o "Biblioteca NetApp" para encontrar a documentação mais atualizada.

Integre o Astra Trident

Para integrar o Astra Trident, os seguintes elementos de design e arquitetura exigem integração: Seleção e implantação de drivers, design de classe de storage, design de pool de storage virtual, impacto na reivindicação de volume persistente (PVC) no

provisionamento de storage, operações de volume e implantação de serviços OpenShift usando o Astra Trident.

Seleção e implantação do driver

Escolha um driver de back-end para ONTAP

Quatro drivers de back-end diferentes estão disponíveis para sistemas ONTAP. Esses drivers são diferenciados pelo protocolo que está sendo usado e como os volumes são provisionados no sistema de storage. Portanto, tenha cuidado em relação ao driver a ser implantado.

Em um nível mais alto, se seu aplicativo tiver componentes que precisam de armazenamento compartilhado (vários pods acessando o mesmo PVC), os drivers baseados em nas seriam a escolha padrão, enquanto os drivers iSCSI baseados em bloco atendem às necessidades de armazenamento não compartilhado. Escolha o protocolo com base nos requisitos da aplicação e no nível de conforto das equipes de armazenamento e infraestrutura. De um modo geral, há pouca diferença entre eles para a maioria dos aplicativos, portanto, muitas vezes a decisão é baseada na necessidade ou não de armazenamento compartilhado (onde mais de um pod precisará de acesso simultâneo).

Os cinco drivers para backends ONTAP estão listados abaixo:

- `ontap-nas`: Cada PV provisionado é um Flexvolume ONTAP completo.
- `ontap-nas-economy`: Cada PV provisionado é uma qtree, com um número configurável de qtrees por Flexvolume (o padrão é 200).
- `ontap-nas-flexgroup`: Cada PV provisionado como um ONTAP FlexGroup completo e todos os agregados atribuídos a um SVM são usados.
- `ontap-san`: Cada PV provisionado é um LUN dentro de seu próprio Flexvolume.
- `ontap-san-economy`: Cada PV provisionado é um LUN, com um número configurável de LUNs por Flexvolume (o padrão é 100).

A escolha entre os três drivers nas tem algumas ramificações para os recursos, que são disponibilizados para o aplicativo.

Observe que, nas tabelas abaixo, nem todos os recursos são expostos pelo Astra Trident. Alguns devem ser aplicados pelo administrador de armazenamento após o provisionamento, se essa funcionalidade for desejada. As notas de rodapé sobrescritas distinguem a funcionalidade por recurso e driver.

Drivers nas ONTAP	Instantâneos	Clones	Políticas de exportação dinâmicas	Ligação múltipla	QoS	Redimensionar	Replicação
`ontap-nas`	Sim	Sim	Nota de rodapé:5[]	Sim	Nota de rodapé:1[]	Sim	Nota de rodapé:1[]
`ontap-nas-economy`	Nota de rodapé:3[]	Nota de rodapé:3[]	Nota de rodapé:5[]	Sim	Nota de rodapé:3[]	Sim	Nota de rodapé:3[]
`ontap-nas-flexgroup`	Nota de rodapé:1[]	Não	Nota de rodapé:5[]	Sim	Nota de rodapé:1[]	Sim	Nota de rodapé:1[]

O Astra Trident oferece 2 drivers SAN para ONTAP, cujas funcionalidades são mostradas abaixo.

Controladores SAN ONTAP	Instantâneos	Clones	Ligação múltipla	CHAP bidirecional	QoS	Redimensionar	Replicação
`ontap-san`	Sim	Sim	Nota de rodapé:4[]	Sim	Nota de rodapé:1[]	Sim	Nota de rodapé:1[]
`ontap-san-economy`	Sim	Sim	Nota de rodapé:4[]	Sim	Nota de rodapé:3[]	Nota de rodapé:1[]	Nota de rodapé:3[]

> Nota de rodapé para as tabelas acima: Yesnote:1[]: Não gerenciado por Astra Trident Yesnote:2[]: Gerenciado por Astra Trident, mas não PV granular Yesnote:3[]: Não gerenciado por Astra Trident e não PV granular Yesnote:4[]: Suportado para volumes em bloco bruto Yesnote:5[]: Suportado por CSI Trident

Os recursos que não são granulares PV são aplicados a todo o Flexvolume e todos os PVS (ou seja, qtrees ou LUNs em FlexVols compartilhados) compartilharão um cronograma comum.

Como podemos ver nas tabelas acima, grande parte da funcionalidade entre `ontap-nas` o e `ontap-nas-economy` é a mesma. No entanto, como o `ontap-nas-economy` motorista limita a capacidade de controlar o cronograma em granularidade por PV, isso pode afetar sua recuperação de desastres e Planejamento de backup em particular. Para as equipes de desenvolvimento que desejam utilizar a funcionalidade de clone de PVC no storage ONTAP, isso só é possível ao usar os `ontap-nas` drivers , `ontap-san` ou `ontap-san-economy` .

> O `solidfire-san` driver também é capaz de clonar PVCs.

Escolha um driver de back-end para Cloud Volumes ONTAP

O Cloud Volumes ONTAP fornece controle de dados junto a recursos de storage de classe empresarial para vários casos de uso, incluindo compartilhamentos de arquivos e storage em nível de bloco, atendendo aos protocolos nas e SAN (NFS, SMB/CIFS e iSCSI). Os drivers compatíveis para o Cloud volume ONTAP são `ontap-nas`, `ontap-nas-economy` `ontap-san` e `ontap-san-economy`. Eles são aplicáveis ao Cloud volume ONTAP para Azure, Cloud volume ONTAP para GCP.

Escolha um driver de back-end para o Amazon FSX for ONTAP

O Amazon FSX for ONTAP permite que os clientes aproveitem os recursos, o desempenho e os recursos administrativos do NetApp com os quais já conhecem, enquanto aproveitam a simplicidade, a agilidade, a segurança e a escalabilidade do armazenamento de dados na AWS. O FSX para ONTAP suporta muitos dos recursos do sistema de arquivos e APIs de administração do ONTAP. Os drivers compatíveis para o Cloud volume ONTAP são `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup` `ontap-san` e `ontap-san-economy`.

Escolha um driver de back-end para NetApp HCI/SolidFire

O `solidfire-san` driver usado com as plataformas NetApp HCI/SolidFire ajuda o administrador a configurar um back-end Element para Trident com base nos limites de QoS. Se você quiser projetar seu back-end para definir os limites de QoS específicos nos volumes provisionados pelo Trident, use o `type` parâmetro no arquivo de back-end. O administrador também pode restringir o tamanho do volume que pode ser criado no

armazenamento usando o `limitVolumeSize` parâmetro. Atualmente, recursos de armazenamento de elementos, como redimensionamento de volume e replicação de volume, não são suportados pelo `solidfire-san` driver. Essas operações devem ser feitas manualmente por meio da IU da Web do Element Software.

Controlador SolidFire	Instantâneos	Clones	Ligação múltipla	CHAP	QoS	Redimensionar	Replicação
`solidfire-san`	Sim	Sim	Nota de rodapé:2[]	Sim	Sim	Sim	Nota de rodapé:1[]

> Nota de rodapé: Yes [1]: Não gerenciado por Astra Trident Yes [2]: Suportado para volumes de blocos brutos

Escolha um driver de back-end para Azure NetApp Files

O Astra Trident usa `azure-netapp-files` o driver para gerenciar o "Azure NetApp Files" serviço.

Mais informações sobre esse driver e como configurá-lo podem ser encontradas no "Configuração de back-end do Astra Trident para Azure NetApp Files".

Controlador Azure NetApp Files	Instantâneos	Clones	Ligação múltipla	QoS	Expandir	Replicação
`azure-netapp-files`	Sim	Sim	Sim	Sim	Sim	Nota de rodapé:1[]

> Nota de rodapé: Yes [1]: Não gerenciado pelo Astra Trident

Escolha um driver de back-end para Cloud Volumes Service com GCP

O Astra Trident usa `gcp-cvs` o driver para vincular ao Cloud Volumes Service no back-end do GCP. Para configurar o back-end do GCP no Trident, é necessário especificar `projectNumber`, `apiRegion` e `apiKey` no arquivo de back-end. O número do projeto pode ser encontrado no portal da Web do GCP, enquanto a chave da API deve ser retirada do arquivo de chave privada da conta de serviço que você criou ao configurar o acesso à API para o Cloud volumes no GCP. O Astra Trident pode criar volumes CVS em um de dois "tipos de serviço":

1. **CVS**: O tipo de serviço CVS básico, que fornece alta disponibilidade por zonas com níveis de desempenho limitados/moderados.
2. **CVS-Performance**: Tipo de serviço otimizado para desempenho mais adequado para cargas de trabalho de produção que valorizam o desempenho. Escolha entre três níveis de serviço exclusivos [`standard`, , `premium` e `extreme`]. Atualmente, o 100 GiB é o tamanho mínimo de volume CVS-Performance que será provisionado, enquanto os volumes CVS devem ser pelo menos 300 GiB. Futuras versões do CVS podem remover essa restrição.

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

Driver CVS para GCP	Instantâneos	Clones	Ligação múltipla	QoS	Expandir	Replicação
`gcp-cvs`	Sim	Sim	Sim	Sim	Sim	Nota de rodapé:1[]

> Nota de rodapé: Yes [1]: Não gerenciado pelo Astra Trident

O `gcp-cvs` driver usa pools de armazenamento virtual. Os pools de storage virtuais abstraem o back-end, permitindo que o Astra Trident decida o posicionamento do volume. O administrador define os pools de armazenamento virtual no(s) arquivo(s) backend.json. As classes de armazenamento identificam os pools de armazenamento virtual com o uso de rótulos.

Design da classe de armazenamento

As classes de armazenamento individuais precisam ser configuradas e aplicadas para criar um objeto Classe de armazenamento Kubernetes. Esta seção discute como projetar uma classe de armazenamento para seu aplicativo.

Design de classe de storage para utilização específica de back-end

A filtragem pode ser usada dentro de um objeto de classe de armazenamento específico para determinar qual pool de armazenamento ou conjunto de pools devem ser usados com essa classe de armazenamento específica. Três conjuntos de filtros podem ser definidos na Classe de armazenamento: `storagePools`, `additionalStoragePools` E/ou `excludeStoragePools`.

O `storagePools` parâmetro ajuda a restringir o armazenamento ao conjunto de pools que correspondem a quaisquer atributos especificados. O `additionalStoragePools` parâmetro é usado para estender o conjunto de pools que o Astra Trident usará para provisionar junto com o conjunto de pools selecionados pelos atributos e `storagePools` parâmetros. Você pode usar um parâmetro sozinho ou ambos juntos para garantir que o conjunto apropriado de pools de armazenamento esteja selecionado.

O `excludeStoragePools` parâmetro é usado para excluir especificamente o conjunto listado de pools que correspondem aos atributos.

Design de classe de storage para emular políticas de QoS

Se você quiser criar classes de armazenamento para emular políticas de qualidade de Serviço, crie uma Classe de armazenamento com o `media` atributo como `hdd` ou `ssd`. Com base no `media` atributo mencionado na classe de storage, o Trident selecionará o back-end apropriado que serve `hdd` ou `ssd` agrega para corresponder ao atributo de Mídia e direcionará o provisionamento dos volumes para o agregado específico. Portanto, podemos criar uma classe de armazenamento PREMIUM que teria um conjunto de atributos, `ssd` que `media` poderia ser classificado como a política de QoS PREMIUM. Podemos criar outro PADRÃO de classe de armazenamento que teria o atributo de Mídia definido como "hdd", que poderia ser classificado como a política de QoS PADRÃO. Também podemos usar o atributo "IOPS" na classe de armazenamento para redirecionar o provisionamento para um dispositivo Element que pode ser definido como uma Política de QoS.

Design de classe de storage para utilizar o back-end com base em recursos específicos

As classes de storage podem ser projetadas para direcionar o provisionamento de volume em um back-end específico, no qual recursos como provisionamento fino e espesso, snapshots, clones e criptografia são ativados. Para especificar qual armazenamento usar, crie classes de armazenamento que especifiquem o

back-end apropriado com o recurso necessário habilitado.

Design de classe de storage para pools de storage virtuais

Os pools de storage virtual estão disponíveis para todos os back-ends Astra Trident. Você pode definir pools de storage virtuais para qualquer back-end, usando qualquer driver fornecido pelo Astra Trident.

Os pools de armazenamento virtual permitem que um administrador crie um nível de abstração sobre backends que pode ser referenciado por meio de classes de armazenamento, para maior flexibilidade e colocação eficiente de volumes em backends. Diferentes backends podem ser definidos com a mesma classe de serviço. Além disso, vários pools de armazenamento podem ser criados no mesmo back-end, mas com caraterísticas diferentes. Quando uma Classe de armazenamento é configurada com um seletor com as etiquetas específicas, o Astra Trident escolhe um back-end que corresponde a todas as etiquetas do seletor para colocar o volume. Se as etiquetas do seletor de classe de storage corresponderem a vários pools de storage, o Astra Trident escolherá um deles para provisionar o volume.

Design de pool de storage virtual

Ao criar um backend, você geralmente pode especificar um conjunto de parâmetros. Era impossível para o administrador criar outro back-end com as mesmas credenciais de armazenamento e com um conjunto diferente de parâmetros. Com a introdução de Virtual Storage Pools, esse problema foi aliviado. O Virtual Storage Pools é uma abstração de nível introduzida entre o back-end e a classe de armazenamento do Kubernetes para que o administrador possa definir parâmetros junto com rótulos que podem ser referenciados por meio das classes de armazenamento do Kubernetes como um seletor, de forma independente de back-end. É possível definir pools de storage virtuais para todos os back-ends NetApp compatíveis com o Astra Trident. Essa lista inclui o SolidFire/NetApp HCI, o ONTAP, o Cloud Volumes Service no GCP e o Azure NetApp Files.

> Ao definir pools de armazenamento virtual, recomenda-se não tentar reorganizar a ordem dos pools virtuais existentes em uma definição de back-end. Também é aconselhável não editar/modificar atributos para um pool virtual existente e definir um novo pool virtual.

Crie pools de storage virtuais para emular diferentes níveis de serviço/QoS

É possível projetar pools de armazenamento virtual para emular classes de serviço. Usando a implementação do pool virtual para o Cloud volume Service for Azure NetApp Files, vamos examinar como podemos configurar diferentes classes de serviço. Configurar o back-end do ANF com várias etiquetas, o que representa diferentes níveis de performance. Defina `servicelevel` Aspect para o nível de desempenho apropriado e adicione outros aspetos necessários em cada rótulo. Agora crie diferentes classes de armazenamento do Kubernetes que mapeariam para diferentes pools de armazenamento virtual. Usando o `parameters.selector` campo, cada StorageClass chama qual(s) pool(s) virtual(s) pode(m) ser usado(s) para hospedar um volume.

Crie pools virtuais para atribuir um conjunto específico de aspetos

Vários pools de storage virtuais com um conjunto específico de aspectos podem ser projetados a partir de um único back-end de storage. Para fazer isso, configure o back-end com vários rótulos e defina os aspetos necessários em cada rótulo. Agora crie diferentes classes de armazenamento do Kubernetes usando o `parameters.selector` campo que mapearia para diferentes pools de armazenamento virtual. Os volumes que são provisionados no back-end terão os aspetos definidos no pool de armazenamento virtual escolhido.

Caraterísticas de PVC que afetam o provisionamento de armazenamento

Alguns parâmetros além da classe de storage solicitada podem afetar o processo de decisão de provisionamento do Astra Trident ao criar uma PVC.

Modo de acesso

Ao solicitar armazenamento através de um PVC, um dos campos obrigatórios é o modo de acesso. O modo desejado pode afetar o back-end selecionado para hospedar a solicitação de armazenamento.

O Astra Trident tentará corresponder ao protocolo de storage usado com o método de acesso especificado de acordo com a matriz a seguir. Isso é independente da plataforma de storage subjacente.

	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
ISCSI	Sim	Sim	Sim (bloco bruto)
NFS	Sim	Sim	Sim

Uma solicitação de um PVC ReadWriteMany enviado para uma implantação do Trident sem um back-end NFS configurado resultará em nenhum volume sendo provisionado. Por esse motivo, o solicitante deve usar o modo de acesso apropriado para sua aplicação.

Operações de volume

Modificar volumes persistentes

Volumes persistentes são, com duas exceções, objetos imutáveis no Kubernetes. Uma vez criados, a política de recuperação e o tamanho podem ser modificados. No entanto, isso não impede que alguns aspectos do volume sejam modificados fora do Kubernetes. Isso pode ser desejável para personalizar o volume para aplicações específicas, para garantir que a capacidade não seja consumida acidentalmente ou simplesmente mover o volume para um controlador de armazenamento diferente por qualquer motivo.

> Atualmente, os provisionadores in-tree do Kubernetes não são compatíveis com operações de redimensionamento de volume para PVS NFS ou iSCSI. O Astra Trident é compatível com a expansão de volumes NFS e iSCSI.

Os detalhes de ligação do PV não podem ser modificados após a criação.

Criar snapshots de volume sob demanda

O Astra Trident é compatível com a criação de snapshot de volume sob demanda e a criação de PVCs a partir de snapshots usando a estrutura CSI. Os snapshots fornecem um método conveniente de manter cópias pontuais dos dados e têm um ciclo de vida independente do PV de origem no Kubernetes. Esses snapshots podem ser usados para clonar PVCs.

Criar volumes a partir de instantâneos

O Astra Trident também suporta a criação de PersistentVolumes a partir de instantâneos de volume. Para conseguir isso, basta criar um PersistentVolumeClaim e mencionar o `datasource` como o instantâneo necessário a partir do qual o volume precisa ser criado. O Astra Trident manipulará esse PVC criando um volume com os dados presentes no snapshot. Com esse recurso, é possível duplicar dados entre regiões, criar ambientes de teste, substituir um volume de produção danificado ou corrompido em sua totalidade, ou recuperar arquivos e diretórios específicos e transferi-los para outro volume anexado.

Mover volumes no cluster

Os administradores de storage podem mover volumes entre agregados e controladores no cluster ONTAP sem interrupções para o consumidor de storage. Essa operação não afeta o Astra Trident nem o cluster Kubernetes, contanto que o agregado de destino seja aquele ao qual o SVM que o Astra Trident está usando tenha acesso. É importante ressaltar que se o agregado tiver sido adicionado recentemente ao SVM, o back-end precisará ser atualizado readicionando-o ao Astra Trident. Isso fará com que o Astra Trident faça o inventário novamente da SVM para que o novo agregado seja reconhecido.

No entanto, a movimentação de volumes entre back-ends não é compatível automaticamente com o Astra Trident. Isso inclui entre SVMs no mesmo cluster, entre clusters ou em uma plataforma de storage diferente (mesmo que esse sistema de storage seja conetado ao Astra Trident).

Se um volume for copiado para outro local, o recurso de importação de volume poderá ser usado para importar volumes atuais para o Astra Trident.

Expanda volumes

O Astra Trident é compatível com o redimensionamento de PVS NFS e iSCSI. Isso permite que os usuários redimensionem seus volumes diretamente pela camada Kubernetes. A expansão de volume é possível para todas as principais plataformas de storage da NetApp, incluindo backends ONTAP, SolidFire/NetApp HCI e Cloud Volumes Service. Para permitir uma possível expansão posterior, defina `allowVolumeExpansion` como `true` no StorageClass associado ao volume. Sempre que for necessário redimensionar o volume persistente, edite a `spec.resources.requests.storage` anotação na reclamação volume persistente para o tamanho de volume pretendido. O Trident cuidará utomaticamente do redimensionamento do volume no cluster de armazenamento.

Importar um volume existente para o Kubernetes

A importação de volume permite importar um volume de storage existente para um ambiente Kubernetes. Atualmente, isso é suportado pelos `ontap-nas` drivers , `ontap-nas-flexgroup`, `solidfire-san`, `azure-netapp-files` e `gcp-cvs` . Esse recurso é útil ao portar um aplicativo existente para o Kubernetes ou durante cenários de recuperação de desastres.

Ao usar o ONTAP e `solidfire-san` os drivers, use o comando `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` para importar um volume existente para o Kubernetes para ser gerenciado pelo Astra Trident. O arquivo de PVC YAML ou JSON usado no comando de volume de importação aponta para uma classe de storage que identifica o Astra Trident como o provisionador. Ao usar um back-end NetApp HCI/SolidFire, certifique-se de que os nomes de volume sejam exclusivos. Se os nomes de volume forem duplicados, clone o volume para um nome exclusivo para que o recurso de importação de volume possa distinguir entre eles.

Se `azure-netapp-files` o driver ou `gcp-cvs` for usado, use o comando `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` para importar o volume para o Kubernetes para ser gerenciado pelo Astra Trident. Isso garante uma referência de volume única.

Quando o comando acima é executado, o Astra Trident encontrará o volume no back-end e lê seu tamanho. Ele irá adicionar automaticamente (e substituir, se necessário) o tamanho de volume do PVC configurado. Em seguida, o Astra Trident cria o novo PV e o Kubernetes liga o PVC ao PV.

Se um recipiente fosse implantado de modo que fosse necessário o PVC importado específico, ele permaneceria em um estado pendente até que o par PVC/PV seja vinculado através do processo de importação de volume. Depois que o par de PVC / PV são ligados, o recipiente deve surgir, desde que não haja outros problemas.

Implantar serviços OpenShift

Os serviços de cluster de valor agregado do OpenShift fornecem funcionalidade importante aos administradores de cluster e aos aplicativos que estão sendo hospedados. O storage que esses serviços usam pode ser provisionado usando os recursos do nó local. No entanto, isso geralmente limita a capacidade, o desempenho, a capacidade de recuperação e a sustentabilidade do serviço. Ao aproveitar um storage array empresarial para fornecer capacidade a esses serviços, é possível melhorar significativamente o serviço. No entanto, como em todas as aplicações, o OpenShift e os administradores de storage devem trabalhar em conjunto para determinar as melhores opções para cada um. A documentação da Red Hat deve ser muito utilizada para determinar os requisitos e garantir que as necessidades de dimensionamento e desempenho sejam atendidas.

Serviço de registo

A implantação e o gerenciamento do armazenamento para o Registro foram documentados "NetApp.io" "blog"no .

Serviço de registo

Assim como outros serviços OpenShift, o serviço de log é implantado usando o Ansible com parâmetros de configuração fornecidos pelo arquivo de inventário, também conhecido como hosts, fornecidos ao manual de estratégia. Há dois métodos de instalação que serão abordados: Implantação de logs durante a instalação inicial do OpenShift e implantação de logs após a instalação do OpenShift.

> A partir do Red Hat OpenShift versão 3,9, a documentação oficial recomenda contra o NFS para o serviço de log devido a preocupações com a corrupção de dados. Isso é baseado no teste da Red Hat de seus produtos. O servidor NFS da ONTAP não tem esses problemas e pode facilmente fazer backup de uma implantação de log. Em última análise, a escolha do protocolo para o serviço de Registro é sua, apenas saiba que ambos funcionarão muito bem ao usar plataformas NetApp e não há motivo para evitar o NFS se essa for sua preferência.

Se você optar por usar o NFS com o serviço de log, precisará definir a variável Ansible `openshift_enable_unsupported_configurations` para `true` evitar que o instalador falhe.

Comece agora

O serviço de log pode, opcionalmente, ser implantado tanto para aplicativos quanto para as operações principais do próprio cluster OpenShift. Se você optar por implantar o Registro de operações, especificando a variável `openshift_logging_use_ops` como `true`, duas instâncias do serviço serão criadas. As variáveis que controlam a instância de log para operações contêm "OPS" nelas, enquanto a instância para aplicativos não.

A configuração das variáveis do Ansible de acordo com o método de implantação é importante para garantir que o storage correto seja utilizado pelos serviços subjacentes. Vamos ver as opções para cada um dos métodos de implantação.

> As tabelas abaixo contêm apenas as variáveis que são relevantes para a configuração de armazenamento, uma vez que se refere ao serviço de registo. Você pode encontrar outras opções nas "Documentação de Registro do RedHat OpenShift" quais devem ser revisadas, configuradas e usadas de acordo com sua implantação.

As variáveis na tabela abaixo resultarão no manual do Ansible criando um PV e PVC para o serviço de Registro usando os detalhes fornecidos. Esse método é significativamente menos flexível do que usar o

manual de instalação de componentes após a instalação do OpenShift, no entanto, se você tiver volumes existentes disponíveis, é uma opção.

Variável	Detalhes
`openshift_logging_storage_kind`	Defina como `nfs` para que o instalador crie um NFS PV para o serviço de log.
`openshift_logging_storage_host`	O nome do host ou endereço IP do host NFS. Isso deve ser definido para o LIF de dados da sua máquina virtual.
`openshift_logging_storage_nfs_directory`	O caminho de montagem para a exportação NFS. Por exemplo, se o volume for juntado como `/openshift_logging`, você usaria esse caminho para essa variável.
`openshift_logging_storage_volume_name`	O nome, por exemplo `pv_ose_logs`, do PV a criar.
`openshift_logging_storage_volume_size`	O tamanho da exportação NFS, por `100Gi` exemplo .

Se o cluster do OpenShift já estiver em execução e, portanto, o Trident tiver sido implantado e configurado, o instalador poderá usar o provisionamento dinâmico para criar os volumes. As variáveis a seguir precisarão ser configuradas.

Variável	Detalhes
`openshift_logging_es_pvc_dynamic`	Defina como verdadeiro para usar volumes provisionados dinamicamente.
`openshift_logging_es_pvc_storage_class_name`	O nome da classe de armazenamento que será usado no PVC.
`openshift_logging_es_pvc_size`	O tamanho do volume solicitado no PVC.
`openshift_logging_es_pvc_prefix`	Um prefixo para os PVCs usados pelo serviço de Registro.
`openshift_logging_es_ops_pvc_dynamic`	Defina como `true` para usar volumes provisionados dinamicamente para a instância de log de operações.
`openshift_logging_es_ops_pvc_storage_class_name`	O nome da classe de armazenamento para a instância de log de operações.
`openshift_logging_es_ops_pvc_size`	O tamanho da solicitação de volume para a instância de operações.
`openshift_logging_es_ops_pvc_prefix`	Um prefixo para os PVCs de instância de OPS.

Implantar a pilha de logs

Se você estiver implantando o log como parte do processo inicial de instalação do OpenShift, então você só precisará seguir o processo de implantação padrão. O Ansible configurará e implantará os serviços necessários e os objetos OpenShift para que o serviço fique disponível assim que o Ansible for concluído.

No entanto, se você estiver implantando após a instalação inicial, o manual de estratégia de componentes precisará ser usado pelo Ansible. Este processo pode mudar ligeiramente com versões diferentes do OpenShift, portanto, certifique-se de ler e seguir "Documentação do RedHat OpenShift Container Platform 3,11" para a sua versão.

Serviço de métricas

O serviço de métricas fornece informações valiosas ao administrador sobre o status, a utilização de recursos e a disponibilidade do cluster OpenShift. Também é necessário para a funcionalidade de escala automática do pod e muitas organizações usam dados do serviço de métricas para seus aplicativos de cobrança e/ou exibição.

Assim como no serviço de log e no OpenShift como um todo, o Ansible é usado para implantar o serviço de métricas. Além disso, tal como o serviço de registo, o serviço de métricas pode ser implementado durante uma configuração inicial do cluster ou depois de estar operacional utilizando o método de instalação do componente. As tabelas a seguir contêm as variáveis que são importantes ao configurar o armazenamento persistente para o serviço de métricas.

As tabelas abaixo contêm apenas as variáveis que são relevantes para a configuração de armazenamento, já que se refere ao serviço de métricas. Há muitas outras opções encontradas na documentação que devem ser revisadas, configuradas e usadas de acordo com sua implantação.

Variável	Detalhes
`openshift_metrics_storage_kind`	Defina como `nfs` para que o instalador crie um NFS PV para o serviço de log.
`openshift_metrics_storage_host`	O nome do host ou endereço IP do host NFS. Isso deve ser definido como o LIF de dados para o SVM.
`openshift_metrics_storage_nfs_directory`	O caminho de montagem para a exportação NFS. Por exemplo, se o volume for juntado como `/openshift_metrics`, você usaria esse caminho para essa variável.
`openshift_metrics_storage_volume_name`	O nome, por exemplo `pv_ose_metrics`, do PV a criar.
`openshift_metrics_storage_volume_size`	O tamanho da exportação NFS, por `100Gi` exemplo .

Se o cluster do OpenShift já estiver em execução e, portanto, o Trident tiver sido implantado e configurado, o instalador poderá usar o provisionamento dinâmico para criar os volumes. As variáveis a seguir precisarão ser configuradas.

Variável	Detalhes
`openshift_metrics_cassandra_pvc_prefix`	Um prefixo a ser usado para as PVCs de métricas.
`openshift_metrics_cassandra_pvc_size`	O tamanho dos volumes a solicitar.
`openshift_metrics_cassandra_storage_type`	O tipo de storage a ser usado para métricas, isso precisa ser definido como dinâmico para que o Ansible crie PVCs com a classe de storage apropriada.
`openshift_metrics_cassanda_pvc_storage_class_name`	O nome da classe de armazenamento a utilizar.

Implantar o serviço de métricas

Com as variáveis apropriadas do Ansible definidas no arquivo de hosts/inventário, implante o serviço com o Ansible. Se você estiver implantando no horário de instalação do OpenShift, o PV será criado e usado automaticamente. Se você estiver implantando usando os playbooks de componentes, após a instalação do OpenShift, o Ansible criará todos os PVCs necessários e, depois que o Astra Trident provisionou o storage para eles, implantará o serviço.

As variáveis acima, e o processo de implantação, podem mudar com cada versão do OpenShift. Certifique-se de rever e seguir "Guia de implantação do OpenShift da RedHat" a sua versão para que ela seja configurada para o seu ambiente.

Proteção de dados

Saiba mais sobre as opções de proteção de dados e capacidade de recuperação que as plataformas de storage da NetApp oferecem. O Astra Trident provisiona volumes que podem aproveitar alguns desses recursos. Você deve ter uma estratégia de proteção e recuperação de dados para cada aplicação com um requisito de persistência.

Faça backup dos `etcd` dados do cluster

O Astra Trident armazena seus metadados no banco de dados do cluster do Kubernetes `etcd`. É importante fazer backup periódico `etcd` dos dados do cluster para recuperar clusters do Kubernetes em cenários de desastre.

Passos

1. O `etcdctl snapshot save` comando permite obter um instantâneo pontual do `etcd` cluster:

   ```
   sudo docker run --rm -v /backup:/backup \
     --network host \
     -v /etc/kubernetes/pki/etcd:/etc/kubernetes/pki/etcd \
     --env ETCDCTL_API=3 \
     k8s.gcr.io/etcd-amd64:3.2.18 \
     etcdctl --endpoints=https://127.0.0.1:2379 \
     --cacert=/etc/kubernetes/pki/etcd/ca.crt \
     --cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt \
     --key=/etc/kubernetes/pki/etcd/healthcheck-client.key \
     snapshot save /backup/etcd-snapshot.db
   ```

   Este comando cria um snapshot etcd girando um contentor etcd e salva-o `/backup` no diretório.

2. No caso de um desastre, você pode aumentar um cluster do Kubernetes usando os snapshots do etcd. Use o `etcdctl snapshot restore` comando para restaurar um instantâneo específico levado para a `/var/lib/etcd` pasta. Depois de restaurar, confirme se a `/var/lib/etcd` pasta foi preenchida com a `member` pasta. O seguinte é um exemplo `etcdctl snapshot restore` de comando:

```
# etcdctl snapshot restore '/backup/etcd-snapshot-latest.db' ; mv
/default.etcd/member/ /var/lib/etcd/
```

3. Antes de inicializar o cluster do Kubernetes, copie todos os certificados necessários.
4. Crie o cluster com o `--ignore-preflight-errors=DirAvailable—var-lib-etcd` sinalizador.
5. Depois que o cluster aparecer, certifique-se de que os pods do sistema kube foram iniciados.
6. Use o `kubectl get crd` comando para verificar se os recursos personalizados criados pelo Trident estão presentes e recuperar objetos Trident para garantir que todos os dados estejam disponíveis.

Recuperar data usando snapshots ONTAP

Os snapshots desempenham um papel importante fornecendo opções de recuperação pontuais para dados de aplicativos. No entanto, os snapshots não são backups sozinhos, eles não protegem contra falhas no sistema de storage ou outras catástrofes. Mas eles são uma maneira conveniente, rápida e fácil de recuperar dados na maioria dos cenários. Saiba mais sobre como usar a tecnologia de snapshot do ONTAP para fazer backups do volume e como restaurá-los.

- Se a política de snapshot não tiver sido definida no back-end, ela será o padrão de uso da `none` política. Isso faz com que o ONTAP não tire snapshots automáticos. No entanto, o administrador de armazenamento pode tirar instantâneos manuais ou alterar a política de instantâneos através da interface de gerenciamento do ONTAP. Isto não afeta o funcionamento do Trident.
- O diretório instantâneo está oculto por padrão. Isso ajuda a facilitar a compatibilidade máxima dos volumes provisionados usando os `ontap-nas` drivers e `ontap-nas-economy`. Ative o `.snapshot` diretório ao usar os `ontap-nas` drivers e `ontap-nas-economy` para permitir que os aplicativos recuperem dados de snapshots diretamente.
- Restaure um volume para um estado gravado em um instantâneo anterior usando o `volume snapshot restore` comando ONTAP CLI. Quando você restaura uma cópia snapshot, a operação de restauração substitui a configuração de volume existente. Todas as alterações feitas aos dados no volume após a criação da cópia Snapshot são perdidas.

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

Replique dados usando o ONTAP

A replicação de dados pode desempenhar um papel importante na proteção contra perda de dados devido a falha do storage array.

Para saber mais sobre as tecnologias de replicação do ONTAP, consulte o "Documentação do ONTAP".

Replicação de máquinas virtuais de storage (SVM) da SnapMirror

Use "SnapMirror" o para replicar um SVM completo, que inclui suas configurações e volumes. Em caso de desastre, você pode ativar o SVM de destino do SnapMirror para começar a fornecer dados. Você pode voltar para o primário quando os sistemas forem restaurados.

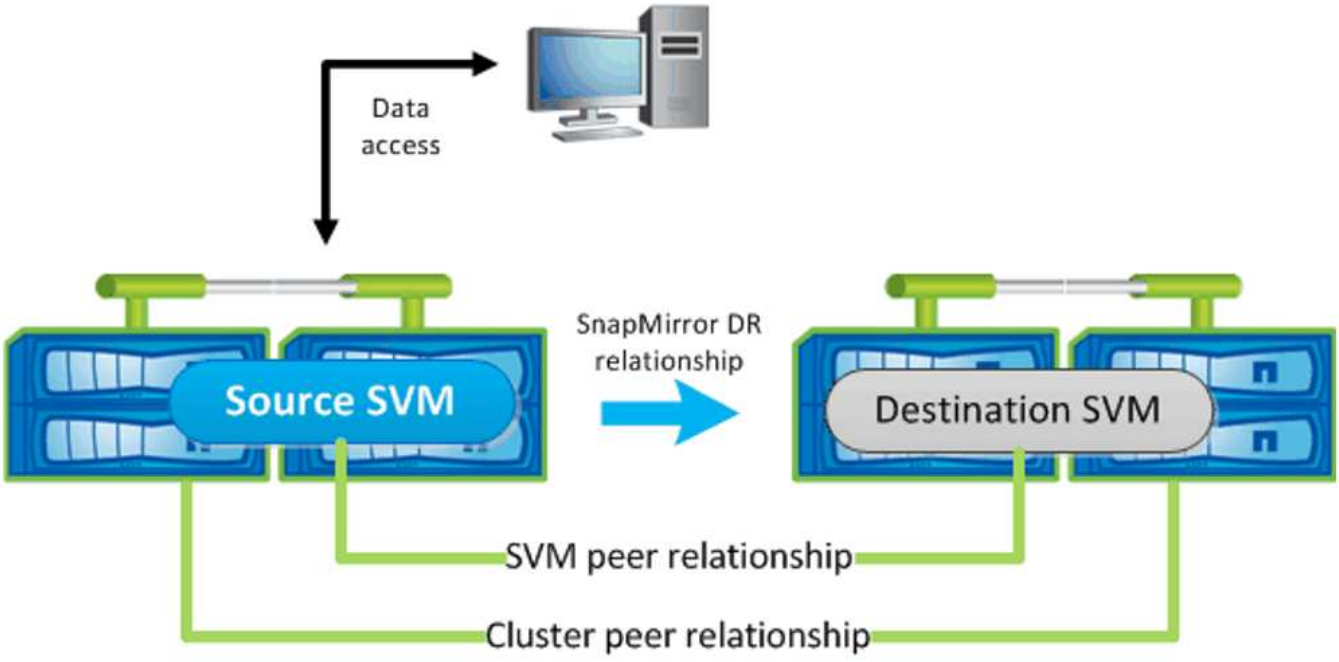
O Astra Trident não pode configurar as relações de replicação por si só. Portanto, o administrador de storage pode usar o recurso replicação do SnapMirror SVM da ONTAP para replicar volumes automaticamente para um destino de recuperação de desastres (DR).

Considere o seguinte caso você esteja planejando usar o recurso replicação do SnapMirror SVM ou esteja usando o recurso atualmente:

- Você deve criar um back-end distinto para cada SVM, que tenha SVM-DR ativado.
- Você deve configurar as classes de armazenamento de modo a não selecionar os backends replicados, exceto quando desejado. Isso é importante para evitar ter volumes que não precisam que a proteção de uma relação de replicação seja provisionada no(s) back(s) que é compatível com a SVM-DR.
- Os administradores de aplicações devem entender o custo e a complexidade adicionais associados à replicação dos dados e um plano de recuperação deve ser determinado antes de utilizar a replicação de dados.
- Antes de ativar o SVM de destino do SnapMirror, interrompa todas as transferências de SnapMirror agendadas, cancele todas as transferências de SnapMirror contínuas, interrompa a relação de replicação, pare a SVM de origem e inicie o SVM de destino do SnapMirror.
- O Astra Trident não detecta automaticamente falhas na SVM. Portanto, após uma falha, o administrador deve executar o `tridentctl backend update` comando para acionar o failover do Trident para o novo back-end.

Aqui está uma visão geral das etapas de configuração da SVM:

- Configure o peering entre o cluster de origem e destino e o SVM.
- Crie o SVM de destino usando a `-subtype dp-destination` opção.
- Crie um agendamento de trabalho de replicação para garantir que a replicação ocorra nos intervalos necessários.
- Crie uma replicação do SnapMirror do SVM de destino para o SVM de origem, usando a `-identity -preserve true` opção para garantir que as configurações de SVM de origem e as interfaces de SVM de origem sejam copiadas para o destino. No SVM de destino, inicialize a relação de replicação do SnapMirror SVM.



Fluxo de trabalho de recuperação de desastres para Trident

O Astra Trident 19,07 e versões posteriores usam CRDs do Kubernetes para armazenar e gerenciar seu próprio estado. Ele usa os clusters do Kubernetes `etcd` para armazenar seus metadados. Aqui assumimos que os arquivos de dados do Kubernetes `etcd` e os certificados são armazenados no NetApp Flexvolume. Esse Flexvolume reside em uma SVM, que tem uma relação SnapMirror SVM-DR com um SVM de destino no local secundário.

As etapas a seguir descrevem como recuperar um único cluster mestre do Kubernetes com o Astra Trident em caso de desastre:

1. Se o SVM de origem falhar, ative o SVM de destino do SnapMirror. Para fazer isso, você deve interromper as transferências agendadas do SnapMirror, cancelar as transferências contínuas do SnapMirror, interromper a relação de replicação, parar o SVM de origem e iniciar o SVM de destino.
2. No SVM de destino, monte o volume que contém os arquivos de dados e certificados do Kubernetes `etcd` no host, que será configurado como um nó mestre.
3. Copie todos os certificados necessários referentes ao cluster do Kubernetes em `/etc/kubernetes/pki` e os arquivos etcd `member` em `/var/lib/etcd`.
4. Crie um cluster do Kubernetes usando o `kubeadm init` comando com o `--ignore-preflight -errors=DirAvailable—var-lib-etcd` sinalizador. Os nomes de host usados para os nós do Kubernetes devem ser os mesmos que o cluster de origem do Kubernetes.
5. Execute o `kubectl get crd` comando para verificar se todos os recursos personalizados do Trident foram criados e recuperar os objetos Trident para verificar se todos os dados estão disponíveis.
6. Atualize todos os backends necessários para refletir o novo nome SVM de destino executando o `./tridentctl update backend <backend-name> -f <backend-json-file> -n <namespace>` comando.

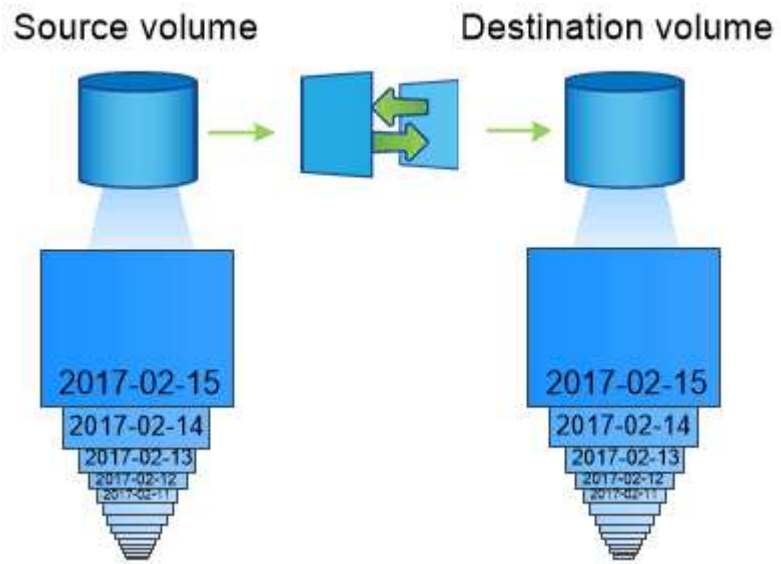
Para volumes persistentes de aplicações, quando o SVM de destino é ativado, todos os volumes provisionados pelo Trident começam a fornecer dados. Depois que o cluster do Kubernetes for configurado no lado do destino usando as etapas descritas acima, todas as implantações e pods são iniciados e as aplicações em contêiner devem ser executadas sem problemas.

Replicação de volume SnapMirror

A replicação de volume ONTAP SnapMirror é um recurso de recuperação de desastres que permite o failover para o storage de destino do storage primário em um nível de volume. O SnapMirror cria uma réplica de volume ou espelhamento do storage primário no storage secundário sincronizando snapshots.

Aqui está uma visão geral das etapas de configuração da replicação de volume do ONTAP SnapMirror:

- Configure o peering entre os clusters nos quais os volumes residem e os SVMs que atendem dados dos volumes.
- Crie uma política SnapMirror, que controla o comportamento da relação e especifica os atributos de configuração para essa relação.
- Crie uma relação SnapMirror entre o volume de destino e o volume de origem usando o[`snapmirror create` comando…] e atribua a política SnapMirror apropriada.
- Depois que a relação SnapMirror for criada, inicialize a relação de modo que uma transferência de linha de base do volume de origem para o volume de destino seja concluída.



Fluxo de trabalho de recuperação de desastres do volume SnapMirror para Trident

As etapas a seguir descrevem como recuperar um único cluster mestre do Kubernetes com o Astra Trident.

1. Em caso de desastre, pare todas as transferências SnapMirror programadas e aborte todas as transferências SnapMirror em curso. Quebre a relação de replicação entre o destino e os volumes de origem para que o volume de destino seja leitura/gravação.
2. No SVM de destino, monte o volume que contém os arquivos de dados e certificados do Kubernetes `etcd` no host, que será configurado como nó principal.

3. Copie todos os certificados necessários referentes ao cluster do Kubernetes em `/etc/kubernetes/pki` e os arquivos etcd `member` em `/var/lib/etcd`.
4. Crie um cluster do Kubernetes executando o `kubeadm init` comando com o `--ignore-preflight-errors=DirAvailable—var-lib-etcd` sinalizador. Os nomes de host devem ser os mesmos que o cluster de origem do Kubernetes.
5. Execute o `kubectl get crd` comando para verificar se todos os recursos personalizados do Trident foram criados e recuperam objetos do Trident para se certificar de que todos os dados estão disponíveis.
6. Limpe os backends anteriores e crie novos backends no Trident. Especifique o novo LIF de dados e gerenciamento, o novo nome da SVM e a senha do SVM de destino.

Fluxo de trabalho de recuperação de desastres para volumes persistentes da aplicação

As etapas a seguir descrevem como os volumes de destino do SnapMirror podem ser disponibilizados para workloads em contêineres em caso de desastre:

1. Pare todas as transferências SnapMirror programadas e aborte todas as transferências SnapMirror em curso. Quebre a relação de replicação entre o destino e o volume de origem para que o volume de destino se torne leitura/gravação. Limpe as implantações que estavam consumindo PVC vinculado a volumes na SVM de origem.
2. Depois que o cluster do Kubernetes for configurado no lado do destino usando as etapas descritas acima, limpe as implantações, PVCs e PV, do cluster do Kubernetes.
3. Crie novos backends no Trident especificando o novo gerenciamento e LIF de dados, o novo nome do SVM e a senha do SVM de destino.
4. Importe os volumes necessários como um PV vinculado a um novo PVC usando o recurso de importação Trident.
5. Reimplante as implantações de aplicativos com os PVCs recém-criados.

Recuperar dados usando snapshots do Element

Faça backup dos dados em um volume de elemento definindo uma programação de instantâneos para o volume e garantindo que os instantâneos sejam obtidos nos intervalos necessários. Você deve definir a programação de snapshot usando a IU ou APIs do Element. Atualmente, não é possível definir um agendamento instantâneo para um volume através `solidfire-san` do controlador.

No caso de corrupção de dados, você pode escolher um snapshot específico e reverter o volume para o snapshot manualmente usando a IU ou APIs do elemento. Isso reverte todas as alterações feitas no volume desde que o snapshot foi criado.

Segurança

Use as recomendações listadas aqui para garantir que sua instalação do Astra Trident esteja segura.

Execute o Astra Trident em seu próprio namespace

É importante impedir que aplicações, administradores de aplicações, usuários e aplicações de gerenciamento acessem as definições de objetos do Astra Trident ou os pods para garantir um storage confiável e bloquear atividades maliciosas em potencial.

Para separar as outras aplicações e usuários do Astra Trident, instale sempre o Astra Trident em seu próprio namespace Kubernetes (`trident`). A colocação do Astra Trident em seu próprio namespace garante que

apenas o pessoal administrativo do Kubernetes tenha acesso ao pod Astra Trident e aos artefatos (como segredos de back-end e CHAP, se aplicável) armazenados nos objetos CRD com namespaces. Você deve garantir que somente os administradores acessem o namespace Astra Trident e, assim, o acesso `tridentctl` à aplicação.

Use a autenticação CHAP com backends ONTAP SAN

O Astra Trident é compatível com autenticação baseada em CHAP para workloads SAN ONTAP (usando os `ontap-san` drivers e `ontap-san-economy`). A NetApp recomenda o uso de CHAP bidirecional com Astra Trident para autenticação entre um host e o back-end de storage.

Para backends ONTAP que usam os drivers de armazenamento SAN, o Astra Trident pode configurar CHAP bidirecional e gerenciar nomes de usuário e segredos do CHAP por meio `tridentctl`do . Veja "aqui" para entender como o Astra Trident configura o CHAP nos backends do ONTAP.

O suporte CHAP para backends ONTAP está disponível com o Trident 20,04 e posterior.

Use a autenticação CHAP com backends NetApp HCI e SolidFire

O NetApp recomenda a implantação de CHAP bidirecional para garantir a autenticação entre um host e os backends NetApp HCI e SolidFire. O Astra Trident usa um objeto secreto que inclui duas senhas CHAP por locatário. Quando o Trident é instalado como um provisionador CSI, ele gerencia os segredos CHAP e os armazena em um `tridentvolume` objeto CR para o respetivo PV. Quando você cria um PV, o CSI Astra Trident usa os segredos CHAP para iniciar uma sessão iSCSI e se comunicar com o sistema NetApp HCI e SolidFire através do CHAP.

Os volumes criados pelo CSI Trident não estão associados a nenhum Grupo de Acesso por volume.

No frontend não-CSI, a vinculação de volumes como dispositivos nos nós de trabalho é tratada pelo Kubernetes. Após a criação de volume, o Astra Trident faz uma chamada de API para o sistema NetApp HCI/SolidFire para recuperar os segredos se o segredo para esse locatário ainda não existir. Em seguida, o Astra Trident passa os segredos para o Kubernetes. O kubelet localizado em cada nó acessa os segredos por meio da API do Kubernetes e os usa para executar/habilitar o CHAP entre cada nó acessando o volume e o sistema NetApp HCI/SolidFire onde os volumes estão localizados.

Referência

Portas Astra Trident

Saiba mais sobre as portas pelas quais o Astra Trident se comunica.

O Astra Trident se comunica pelas seguintes portas:

Porta	Finalidade
8443	Backchannel HTTPS
8001	Endpoint de métricas Prometheus
8000	SERVIDOR REST do Trident
17546	Porta de sonda de disponibilidade/disponibilidade usada pelos pods daemonset Trident

A porta da sonda de disponibilidade/disponibilidade pode ser alterada durante o tempo de instalação utilizando o `--probe-port` sinalizador. É importante garantir que essa porta não esteja sendo usada por outro processo nos nós de trabalho.

API REST do Astra Trident

"comandos e opções tridentctl"Embora seja a maneira mais fácil de interagir com a API REST do Astra Trident, você pode usar o endpoint REST diretamente se preferir.

Isso é útil para instalações avançadas que usam o Astra Trident como um binário autônomo em implantações não Kubernetes.

Para uma melhor segurança, o Astra Trident's `REST API` é restrito a localhost por padrão ao ser executado dentro de um pod. Para alterar esse comportamento, você precisa definir o argumento do Astra Trident `-address` na configuração do pod.

A API funciona da seguinte forma:

`GET`

- `GET <trident-address>/trident/v1/<object-type>`: Lista todos os objetos desse tipo.
- `GET <trident-address>/trident/v1/<object-type>/<object-name>`: Obtém os detalhes do objeto nomeado.

`POST`

`POST <trident-address>/trident/v1/<object-type>`: Cria um objeto do tipo especificado.

- Requer uma configuração JSON para o objeto a ser criado. Para a especificação de cada tipo de objeto, consulte `tridentctl` comandos e opções.
- Se o objeto já existir, o comportamento varia: Os backends atualizam o objeto existente, enquanto todos

os outros tipos de objeto falharão a operação.

`DELETE`

`DELETE <trident-address>/trident/v1/<object-type>/<object-name>`: Exclui o recurso nomeado.

> Os volumes associados a backends ou classes de armazenamento continuarão a existir; estes devem ser excluídos separadamente. Para obter mais informações, consulte o `tridentctl` comandos e opções.

Para exemplos de como essas APIs são chamadas, passe o (`-d`sinalizador debug). Para obter mais informações, consulte o `tridentctl` comandos e opções.

Opções de linha de comando

O Astra Trident expõe várias opções de linha de comando para o orquestrador Trident. Você pode usar essas opções para modificar sua implantação.

A registar

- `-debug`: Ativa a saída de depuração.
- `-loglevel <level>`: Define o nível de log (debug, info, warn, error, fatal). O padrão é INFO.

Kubernetes

- `-k8s_pod`: Use esta opção ou `-k8s_api_server` para ativar o suporte do Kubernetes. Isso faz com que o Trident use suas credenciais de conta de serviço do Kubernetes do pod que contêm para entrar em Contato com o servidor de API. Isso só funciona quando o Trident é executado como um pod em um cluster do Kubernetes com contas de serviço ativadas.
- `-k8s_api_server <insecure-address:insecure-port>`: Use esta opção ou `-k8s_pod` para ativar o suporte do Kubernetes. Quando especificado, o Trident se coneta ao servidor de API do Kubernetes usando o endereço e a porta inseguros fornecidos. Isso permite que o Trident seja implantado fora de um pod; no entanto, ele só suporta conexões inseguras com o servidor de API. Para se conetar com segurança, implante o Trident em um pod com a `-k8s_pod` opção.
- `-k8s_config_path <file>`: Obrigatório; você deve especificar esse caminho para um arquivo KubeConfig.

Docker

- `-volume_driver <name>`: Nome do driver usado ao Registrar o plug-in do Docker. O padrão é `netapp`.
- `-driver_port <port-number>`: Ouça nesta porta em vez de um soquete de domínio UNIX.
- `-config <file>`: Obrigatório; você deve especificar esse caminho para um arquivo de configuração de back-end.

DESCANSO

- `-address <ip-or-host>`: Especifica o endereço no qual o servidor REST do Trident deve escutar. O padrão é localhost. Ao ouvir no localhost e executar dentro de um pod Kubernetes, a interface REST não é diretamente acessível de fora do pod. `-address ""`Utilize para tornar a INTERFACE REST acessível a partir do endereço IP do pod.

> A interface REST DO Trident pode ser configurada para ouvir e servir apenas em 127.0.0.1 (para IPv4) ou [::1] (para IPv6).

- `-port <port-number>`: Especifica a porta na qual o servidor REST do Trident deve escutar. O padrão é 8000.
- `-rest`: Ativa a interface REST. O padrão é verdadeiro.

Produtos da NetApp integrados ao Kubernetes

O portfólio de produtos de storage do NetApp se integra a muitos aspectos diferentes de um cluster Kubernetes, fornecendo recursos avançados de gerenciamento de dados que melhoram o recurso, a funcionalidade, a performance e a disponibilidade da implantação do Kubernetes.

Astra

"Astra" Facilita o gerenciamento, a proteção e a migração de workloads em contêineres com muitos dados executados no Kubernetes dentro e entre nuvens públicas e no local. O Astra provisiona e fornece storage de contêiner persistente usando o Trident do portfólio de storage comprovado e expansivo da NetApp, na nuvem pública e no local. Ele também oferece um conjunto avançado de recursos avançados de gerenciamento de dados com reconhecimento de aplicações, como snapshot, backup e restauração, logs de atividade e clonagem ativa para proteção de dados, recuperação de desastres/dados, auditoria de dados e casos de uso de migração para workloads Kubernetes.

ONTAP

O ONTAP é o sistema operacional de storage unificado multiprotocolo da NetApp que oferece recursos avançados de gerenciamento de dados para qualquer aplicação. Os sistemas ONTAP têm configurações all-flash, híbridas ou totalmente HDD e oferecem muitos modelos de implantação diferentes, incluindo hardware projetado (FAS e AFF), white-box (ONTAP Select) e somente para nuvem (Cloud Volumes ONTAP).

> O Trident oferece suporte a todos os modelos de implantação do ONTAP mencionados acima.

Cloud Volumes ONTAP

"Cloud Volumes ONTAP" É um dispositivo de storage somente de software que executa o software de gerenciamento de dados ONTAP na nuvem. Use o Cloud Volumes ONTAP para workloads de produção, recuperação de desastres, DevOps, compartilhamentos de arquivos e gerenciamento de banco de dados. Ele amplia o storage empresarial para a nuvem oferecendo eficiência de storage, alta disponibilidade, replicação de dados, disposição em categorias e consistência de aplicações.

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 ONTAP da

NetApp. O FSX para ONTAP permite que os clientes aproveitem os recursos, o desempenho e os recursos administrativos do NetApp com os quais já conhecem, enquanto aproveitam a simplicidade, a agilidade, a segurança e a escalabilidade do armazenamento de dados na AWS. O FSX para ONTAP suporta muitos dos recursos do sistema de arquivos e APIs de administração do ONTAP.

Software Element

"Elemento" permite que o administrador de storage consolide workloads garantindo a performance e possibilitando um espaço físico do storage simplificado e otimizado. Acoplado a uma API que permite a automação de todos os aspectos do gerenciamento de storage, o Element permite que os administradores de storage façam mais com menos esforço.

NetApp HCI

"NetApp HCI" simplifica o gerenciamento e a escala do data center automatizando tarefas de rotina e permitindo que os administradores de infraestrutura se concentrem em funções mais importantes.

O NetApp HCI é totalmente suportado pelo Trident. O Trident pode provisionar e gerenciar dispositivos de storage para aplicações em contêiner diretamente na plataforma de storage subjacente da NetApp HCI.

Azure NetApp Files

"Azure NetApp Files" É um serviço de compartilhamento de arquivos do Azure de nível empresarial, desenvolvido pela NetApp. É possível executar os workloads mais exigentes baseados em arquivos no Azure de forma nativa, com a performance e o gerenciamento de rich data que você espera do NetApp.

Cloud Volumes Service para Google Cloud

"NetApp Cloud Volumes Service para Google Cloud" É um serviço de arquivos nativo da nuvem que fornece volumes nas em NFS e SMB com performance all-flash. Esse serviço permite que qualquer workload, incluindo aplicações legadas, seja executado na nuvem do GCP. Ele fornece um serviço totalmente gerenciado que oferece alta performance consistente, clonagem instantânea, proteção de dados e acesso seguro às instâncias do Google Compute Engine (GCE).

Objetos Kubernetes e Trident

É possível interagir com o Kubernetes e o Trident usando APIs REST lendo e escrevendo objetos de recursos. Há vários objetos de recursos que ditam a relação entre o Kubernetes e o Trident, o Trident e o storage, o Kubernetes e o storage. Alguns desses objetos são gerenciados pelo Kubernetes e os outros são gerenciados pelo Trident.

Como os objetos interagem uns com os outros?

Talvez a maneira mais fácil de entender os objetos, para que eles são e como eles interagem seja seguir uma única solicitação de armazenamento de um usuário do Kubernetes:

1. Um usuário cria `PersistentVolumeClaim` uma solicitação de um novo `PersistentVolume` de um tamanho específico de um Kubernetes `StorageClass` que foi configurado anteriormente pelo administrador.
2. O Kubernetes `StorageClass` identifica o Trident como seu provisionador e inclui parâmetros que informam ao Trident como provisionar um volume para a classe solicitada.

3. O Trident olha para si `StorageClass` mesmo com o mesmo nome que identifica a correspondência `Backends` e `StoragePools` que pode usar para provisionar volumes para a classe.
4. O Trident provisiona o storage em um back-end compatível e cria dois objetos: Um `PersistentVolume` no Kubernetes que diz ao Kubernetes como encontrar, montar e tratar o volume e um volume no Trident que mantém a relação entre o `PersistentVolume` e o storage real.
5. O Kubernetes vincula `PersistentVolumeClaim` o ao novo `PersistentVolume`. Pods que incluem a `PersistentVolumeClaim` montagem que Persistentvolume em qualquer host em que ele seja executado.
6. Um usuário cria um `VolumeSnapshot` de um PVC existente, usando um `VolumeSnapshotClass` que aponta para Trident.
7. O Trident identifica o volume que está associado ao PVC e cria um snapshot do volume em seu back-end. Ele também cria um `VolumeSnapshotContent` que instrui o Kubernetes sobre como identificar o snapshot.
8. Um usuário pode criar um `PersistentVolumeClaim` usando `VolumeSnapshot` como fonte.
9. O Trident identifica o instantâneo necessário e executa o mesmo conjunto de etapas envolvidas na criação de um `PersistentVolume` e um `Volume`.

Para ler mais sobre objetos do Kubernetes, é altamente recomendável que você leia a "Volumes persistentes" seção da documentação do Kubernetes.

Objetos do Kubernetes `PersistentVolumeClaim`

Um objeto Kubernetes `PersistentVolumeClaim` é uma solicitação de storage feita por um usuário de cluster do Kubernetes.

Além da especificação padrão, o Trident permite que os usuários especifiquem as seguintes anotações específicas de volume se quiserem substituir os padrões definidos na configuração de back-end:

Anotação	Opção de volume	Drivers suportados
Trident.NetApp.io/sistema de arquivos	Sistema de ficheiros	ONTAP-san, SolidFire-san, eseries-iscsi, ONTAP-san-economy
Trident.NetApp.io/cloneFromPVC	CloneSourcevolume	ONTAP-nas, ONTAP-san, SolidFire-san, azure-NetApp-files, gcp-cvs, ONTAP-san-economy
Trident.NetApp.io/splitOnClone	SplitOnClone	ONTAP-nas, ONTAP-san
Trident.NetApp.io/protocolo	protocolo	qualquer
Trident.NetApp.io/exportPolicy	Política de exportação	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup
Trident.NetApp.io/snapshotPolicy	Política de SnapshotPolicy	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san
Trident.NetApp.io/snapshotServe	SnapshotServe	ONTAP-nas, ONTAP-nas-FlexGroup, ONTAP-san, gcp-cvs

Anotação	Opção de volume	Drivers suportados
Trident.NetApp.io/snapshotDirectory	SnapshotDirectory	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup
Trident.NetApp.io/unixPermissions	UnixPermissions	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup
Trident.NetApp.io/blocksize	Tamanho do bloco	SolidFire-san

Se o PV criado tiver a `Delete` política de recuperação, o Trident excluirá o PV e o volume de backup quando o PV for liberado (ou seja, quando o usuário exclui o PVC). Caso a ação de exclusão falhe, o Trident marca o PV como tal e periodicamente tenta novamente a operação até que seja bem-sucedida ou o PV seja excluído manualmente. Se o PV usar a `Retain` política, o Trident a ignora e assume que o administrador irá limpá-la do Kubernetes e do back-end, permitindo que o volume seja feito backup ou inspecionado antes de sua remoção. Observe que a exclusão do PV não faz com que o Trident exclua o volume de backup. Você deve removê-lo usando a API REST (`tridentctl`).

O Trident dá suporte à criação de snapshots de volume usando a especificação CSI: Você pode criar um instantâneo de volume e usá-lo como fonte de dados para clonar PVCs existentes. Dessa forma, cópias pontuais de PVS podem ser expostas ao Kubernetes na forma de snapshots. Os instantâneos podem então ser usados para criar novos PVS. Dê uma olhada `On-Demand Volume Snapshots` para ver como isso funcionaria.

O Trident também fornece as `cloneFromPVC` anotações e `splitOnClone` para a criação de clones. Você pode usar essas anotações para clonar um PVC sem precisar usar a implementação do CSI (no Kubernetes 1,13 e anterior) ou se sua versão do Kubernetes não for compatível com snapshots de volume beta (Kubernetes 1,16 e anteriores). Tenha em mente que o Trident 19,10 suporta o fluxo de trabalho CSI para clonagem a partir de um PVC.

> Você pode usar as `cloneFromPVC` anotações e `splitOnClone` com o CSI Trident, bem como o frontend tradicional não CSI.

Aqui está um exemplo: Se um usuário já tem um PVC chamado `mysql`, o usuário pode criar um novo PVC chamado `mysqlclone` usando a anotação, como `trident.netapp.io/cloneFromPVC: mysql` . Com esse conjunto de anotações, o Trident clona o volume correspondente ao PVC mysql, em vez de provisionar um volume do zero.

Considere os seguintes pontos:

- Recomendamos clonar um volume ocioso.
- O PVC e seu clone devem estar no mesmo namespace do Kubernetes e ter a mesma classe de storage.
- Com os `ontap-nas` drivers e `ontap-san`, pode ser desejável definir a anotação de PVC `trident.netapp.io/splitOnClone` em conjunto `trident.netapp.io/cloneFromPVC` com o . Com `trident.netapp.io/splitOnClone` definido como `true`, o Trident divide o volume clonado do volume pai e, portanto, desacoplando completamente o ciclo de vida do volume clonado de seus pais às custas de perder alguma eficiência de storage. Não `trident.netapp.io/splitOnClone` configurá-lo ou configurá-lo para `false` resultar em consumo de espaço reduzido no back-end à custa de criar dependências entre os volumes pai e clone, de modo que o volume pai não possa ser excluído a menos que o clone seja excluído primeiro. Um cenário em que dividir o clone faz sentido é clonar um volume de banco de dados vazio, onde é esperado que o volume e seu clone diverjam muito e não se beneficiem das eficiências de armazenamento oferecidas pelo ONTAP.

O `sample-input` diretório contém exemplos de definições de PVC para uso com Trident. Consulte objetos de volume do Trident para obter uma descrição completa dos parâmetros e configurações associados aos volumes do Trident.

Objetos do Kubernetes `PersistentVolume`

Um objeto Kubernetes `PersistentVolume` representa um storage disponibilizado para o cluster do Kubernetes. Ele tem um ciclo de vida que é independente do pod que o usa.

> O Trident cria `PersistentVolume` objetos e os Registra no cluster do Kubernetes automaticamente com base nos volumes provisionados. Você não é esperado para gerenciá-los sozinho.

Quando você cria um PVC que se refere a um Trident-based `StorageClass`, o Trident provisiona um novo volume usando a classe de armazenamento correspondente e Registra um novo PV para esse volume. Ao configurar o volume provisionado e o PV correspondente, o Trident segue as seguintes regras:

- O Trident gera um nome PV para o Kubernetes e um nome interno que ele usa para provisionar o storage. Em ambos os casos, é garantir que os nomes são únicos em seu escopo.
- O tamanho do volume corresponde ao tamanho solicitado no PVC o mais próximo possível, embora possa ser arredondado para a quantidade alocável mais próxima, dependendo da plataforma.

Objetos do Kubernetes `StorageClass`

Os objetos Kubernetes `StorageClass` são especificados por nome em `PersistentVolumeClaims` para provisionar o storage com um conjunto de propriedades. A própria classe de storage identifica o provisionador a ser usado e define esse conjunto de propriedades em termos que o provisionador entende.

É um dos dois objetos básicos que precisam ser criados e gerenciados pelo administrador. O outro é o objeto backend do Trident.

Um objeto do Kubernetes `StorageClass` que usa o Trident é parecido com este:

```
apiVersion: storage.k8s.io/v1beta1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters:
  <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Esses parâmetros são específicos do Trident e informam à Trident como provisionar volumes para a classe.

Os parâmetros da classe de armazenamento são:

Atributo	Tipo	Obrigatório	Descrição
atributos	map[string]string	não	Veja a seção atributos abaixo
StoragePools	MAP[string]StringList	não	Mapa de nomes de back-end para listas de pools de armazenamento dentro
Além disso, StoragePools	MAP[string]StringList	não	Mapa de nomes de back-end para listas de pools de armazenamento dentro
Excluir StoragePools	MAP[string]StringList	não	Mapa de nomes de back-end para listas de pools de armazenamento dentro

Os atributos de storage e seus possíveis valores podem ser classificados em atributos de seleção de pool de storage e atributos do Kubernetes.

Atributos de seleção do pool de armazenamento

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

Atributo	Tipo	Valores	Oferta	Pedido	Suportado por
1	cadeia de carateres	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 carateres	fino, grosso	O pool é compatível com esse método de provisionamento	Método de provisionamento especificado	espessura: todos os ONTAP e séries-iscsi; fino: todos os ONTAP e SolidFire-san
BackendType	cadeia de carateres	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san, SolidFire-san, eseries-iscsi, gcp-cvs, azure-NetApp-files, ONTAP-san-economy	Pool pertence a este tipo de backend	Back-end especificado	Todos os drivers

Atributo	Tipo	Valores	Oferta	Pedido	Suportado por
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

Na maioria dos casos, os valores solicitados influenciam diretamente o provisionamento; por exemplo, a solicitação de provisionamento espesso resulta em um volume provisionado rapidamente. No entanto, um pool de storage de elemento usa o mínimo e o máximo de IOPS oferecidos para definir valores de QoS, em vez do valor solicitado. Nesse caso, o valor solicitado é usado apenas para selecionar o pool de armazenamento.

O ideal é usar `attributes` sozinho para modelar as qualidades do storage de que você precisa para atender às necessidades de uma classe específica. O Trident deteta e seleciona automaticamente pools de armazenamento que correspondem a *all* do `attributes` que você especificar.

Se você não conseguir usar `attributes` para selecionar automaticamente os pools certos para uma classe, use os `storagePools` parâmetros e `additionalStoragePools` para refinar ainda mais os pools ou até mesmo selecionar um conjunto específico de pools.

Você pode usar o `storagePools` parâmetro para restringir ainda mais o conjunto de pools que correspondem a qualquer `attributes` especificado . Em outras palavras, o Trident usa a interseção de pools identificados pelos `attributes` parâmetros e `storagePools` para o provisionamento. Você pode usar um parâmetro sozinho ou ambos juntos.

Você pode usar o `additionalStoragePools` parâmetro para estender o conjunto de pools que o Trident usa para provisionamento, independentemente de quaisquer pools selecionados pelos `attributes` parâmetros e. `storagePools`

Você pode usar o `excludeStoragePools` parâmetro para filtrar o conjunto de pools que o Trident usa para provisionar. O uso desse parâmetro remove todos os pools que correspondem.

`storagePools`Nos parâmetros e `additionalStoragePools`, cada entrada assume o formulário `<backend>:<storagePoolList>`, onde `<storagePoolList>` é uma lista separada por vírgulas de pools de armazenamento para o back-end especificado. Por exemplo, um valor para `additionalStoragePools` pode parecer como `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`. Essas listas aceitam valores de regex tanto para os valores de backend quanto de lista. Você pode usar `tridentctl get backend` para obter a lista de backends e suas piscinas.

Atributos do Kubernetes

Esses atributos não têm impacto na seleção de pools de storage/back-ends pelo Trident durante o provisionamento dinâmico. Em vez disso, esses atributos simplesmente fornecem parâmetros compatíveis com volumes persistentes do Kubernetes. Os nós de trabalho são responsáveis pelas operações de criação de sistema de arquivos e podem exigir utilitários de sistema de arquivos, como xfsprogs.

Atributo	Tipo	Valores	Descrição	Drivers relevantes	Versão do Kubernetes
FsType	cadeia de carateres	ext4, ext3, xfs, etc.	O tipo de sistema de arquivos para volumes de bloco	SolidFire-san, ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san, ONTAP-san-economy, eseries-iscsi	Tudo
AllowVolumeExpansion	booleano	verdadeiro, falso	Ative ou desative o suporte para aumentar o tamanho do PVC	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san, ONTAP-san-economy, SolidFire-san, gcp-cvs, azure-NetApp-files	Mais de 1,11 anos
VolumeBindingMode	cadeia de carateres	Imediato, WaitForFirstConsumer	Escolha quando ocorre a vinculação de volume e o provisionamento dinâmico	Tudo	1,18 - 1,24

- O `fsType` parâmetro é usado para controlar o tipo de sistema de arquivos desejado para LUNs SAN. Além disso, o Kubernetes também usa a presença de `fsType` em uma classe de armazenamento para indicar que existe um sistema de arquivos. A propriedade do volume só pode ser controlada usando o `fsGroup` contexto de segurança de um pod se `fsType` estiver definido. Consulte "Kubernetes: Configurar um contexto de segurança para um pod ou contêiner" para obter uma visão geral sobre como definir a propriedade do volume usando o `fsGroup` contexto. O Kubernetes aplicará o `fsGroup` valor somente se:

  - `fsType` é definido na classe de armazenamento.
  - O modo de acesso de PVC é RWO.

  Para drivers de armazenamento NFS, já existe um sistema de arquivos como parte da exportação NFS. Para usar `fsGroup` a classe de armazenamento ainda precisa especificar um `fsType`. você pode configurá-lo como `nfs` ou qualquer valor não nulo.

- "Expanda volumes"Consulte para obter mais detalhes sobre a expansão de volume.
- O pacote de instalação do Trident fornece vários exemplos de definições de classe de armazenamento para uso com o Trident no `sample-input/storage-class-*.yaml`. A exclusão de uma classe de armazenamento Kubernetes faz com que a classe de armazenamento Trident correspondente também seja excluída.

Objetos do Kubernetes `VolumeSnapshotClass`

Os objetos do Kubernetes `VolumeSnapshotClass` são análogos ao `StorageClasses`. Eles ajudam a definir várias classes de armazenamento e são referenciados por instantâneos de volume para associar o snapshot à classe de snapshot necessária. Cada snapshot de volume é associado a uma classe de snapshot de volume único.

A `VolumeSnapshotClass` deve ser definida por um administrador para criar instantâneos. Uma classe de instantâneo de volume é criada com a seguinte definição:

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

O `driver` especifica ao Kubernetes que as solicitações de snapshots de volume `csi-snapclass` da classe são tratadas pelo Trident. O `deletionPolicy` especifica a ação a ser tomada quando um instantâneo deve ser excluído. `deletionPolicy`Quando está definido como `Delete`, os objetos instantâneos de volume e o instantâneo subjacente no cluster de armazenamento são removidos quando um instantâneo é excluído. Alternativamente, configurá-lo para `Retain` significa que `VolumeSnapshotContent` e o instantâneo físico são retidos.

Objetos do Kubernetes `VolumeSnapshot`

Um objeto Kubernetes `VolumeSnapshot` é uma solicitação para criar um snapshot de um volume. Assim como um PVC representa uma solicitação feita por um usuário para um volume, um instantâneo de volume é

uma solicitação feita por um usuário para criar um instantâneo de um PVC existente.

Quando uma solicitação de snapshot de volume entra, o Trident gerencia automaticamente a criação do snapshot para o volume no back-end e expõe o snapshot criando um objeto exclusivo `VolumeSnapshotContent`. Você pode criar snapshots a partir de PVCs existentes e usar os snapshots como DataSource ao criar novos PVCs.

> A vida útil de um VolumeSnapshot é independente do PVC de origem: Um snapshot persiste mesmo depois que o PVC de origem é excluído. Ao excluir um PVC que tenha instantâneos associados, o Trident marca o volume de apoio para este PVC em um estado **Deletando**, mas não o remove completamente. O volume é removido quando todos os instantâneos associados são excluídos.

Objetos do Kubernetes `VolumeSnapshotContent`

Um objeto Kubernetes `VolumeSnapshotContent` representa um snapshot retirado de um volume já provisionado. Ele é análogo a `PersistentVolume` e significa um snapshot provisionado no cluster de storage. Semelhante aos `PersistentVolumeClaim` objetos e `PersistentVolume`, quando um snapshot é criado, o `VolumeSnapshotContent` objeto mantém um mapeamento um-para-um para o `VolumeSnapshot` objeto, que havia solicitado a criação do snapshot.

> O Trident cria `VolumeSnapshotContent` objetos e os Registra no cluster do Kubernetes automaticamente com base nos volumes provisionados. Você não é esperado para gerenciá-los sozinho.

O `VolumeSnapshotContent` objeto contém detalhes que identificam exclusivamente o instantâneo, como o `snapshotHandle`. Esta `snapshotHandle` é uma combinação única do nome do PV e do nome do `VolumeSnapshotContent` objeto.

Quando uma solicitação de snapshot entra, o Trident cria o snapshot no back-end. Depois que o snapshot é criado, o Trident configura um `VolumeSnapshotContent` objeto e, portanto, expõe o snapshot à API do Kubernetes.

Objetos do Kubernetes `CustomResourceDefinition`

Os recursos personalizados do Kubernetes são endpoints na API do Kubernetes que são definidos pelo administrador e são usados para agrupar objetos semelhantes. O Kubernetes dá suporte à criação de recursos personalizados para armazenar uma coleção de objetos. Você pode obter essas definições de recursos executando `kubectl get crds`o .

As definições personalizadas de recursos (CRDs) e os metadados de objetos associados são armazenados pelo Kubernetes em seu armazenamento de metadados. Isso elimina a necessidade de uma loja separada para o Trident.

A partir da versão 19,07, o Trident usa vários `CustomResourceDefinition` objetos para preservar a identidade de objetos Trident, como backends Trident, classes de armazenamento Trident e volumes Trident. Esses objetos são gerenciados pelo Trident. Além disso, a estrutura de snapshot do volume CSI introduz algumas CRDs que são necessárias para definir snapshots de volume.

CRDs são uma construção do Kubernetes. Os objetos dos recursos definidos acima são criados pelo Trident. Como um exemplo simples, quando um back-end é criado usando `tridentctl`o , um objeto CRD correspondente `tridentbackends` é criado para consumo pelo Kubernetes.

Aqui estão alguns pontos a ter em mente sobre os CRDs do Trident:

- Quando o Trident é instalado, um conjunto de CRDs é criado e pode ser usado como qualquer outro tipo de recurso.
- Ao atualizar a partir de uma versão anterior do Trident (uma que usou `etcd` para manter o estado), o instalador do Trident migra dados do `etcd` armazenamento de dados de valor-chave e cria objetos CRD correspondentes.
- Ao desinstalar o Trident usando o `tridentctl uninstall` comando, os pods Trident são excluídos, mas os CRDs criados não são limpos. Veja "Desinstale o Trident" para entender como o Trident pode ser completamente removido e reconfigurado do zero.

ObjetosTrident `StorageClass`

O Trident cria classes de storage correspondentes para objetos Kubernetes `StorageClass` que especificam `csi.trident.netapp.io/netapp.io/trident` no campo do provisionador. O nome da classe de storage corresponde ao do objeto Kubernetes `StorageClass` que ele representa.

> Com o Kubernetes, esses objetos são criados automaticamente quando um Kubernetes `StorageClass` que usa o Trident como provisionador é registrado.

As classes de armazenamento compreendem um conjunto de requisitos para volumes. O Trident atende a esses requisitos com os atributos presentes em cada pool de storage. Se forem correspondentes, esse pool de storage será um destino válido para volumes de provisionamento que usam essa classe de storage.

Você pode criar configurações de classe de armazenamento para definir diretamente classes de armazenamento usando a API REST. No entanto, para implantações do Kubernetes, esperamos que elas sejam criadas ao Registrar novos objetos do Kubernetes `StorageClass`.

Objetos de back-end do Trident

Os backends representam os fornecedores de storage em cima dos quais o Trident provisiona volumes. Uma única instância do Trident pode gerenciar qualquer número de backends.

> Este é um dos dois tipos de objetos que você cria e gerencia a si mesmo. O outro é o objeto Kubernetes `StorageClass`.

Para obter mais informações sobre como construir esses objetos, consulte Configuração de back-end.

ObjetosTrident `StoragePool`

Os pools de storage representam locais distintos disponíveis para provisionamento em cada back-end. Para ONTAP, eles correspondem a agregados em SVMs. Para NetApp HCI/SolidFire, estes correspondem a bandas de QoS especificadas pelo administrador. Para o Cloud Volumes Service, eles correspondem a regiões de provedores de nuvem. Cada pool de storage tem um conjunto de atributos de storage distintos, que definem suas características de performance e proteção de dados.

Ao contrário dos outros objetos aqui, os candidatos ao pool de armazenamento são sempre descobertos e gerenciados automaticamente.

ObjetosTrident `Volume`

Os volumes são a unidade básica de provisionamento, incluindo pontos de extremidade de back-end, como compartilhamentos NFS e iSCSI LUNs. No Kubernetes, eles correspondem diretamente `PersistentVolumes` ao . Ao criar um volume, certifique-se de que ele tenha uma classe de armazenamento, que determina onde esse volume pode ser provisionado, juntamente com um tamanho.

No Kubernetes, esses objetos são gerenciados automaticamente. Você pode visualizá-los para ver o que o Trident provisionou.

Ao excluir um PV com instantâneos associados, o volume Trident correspondente é atualizado para um estado **Deletando**. Para que o volume Trident seja excluído, você deve remover os snapshots do volume.

Uma configuração de volume define as propriedades que um volume provisionado deve ter.

Atributo	Tipo	Obrigatório	Descrição
versão	cadeia de carateres	não	Versão da API Trident ("1")
nome	cadeia de carateres	sim	Nome do volume a criar
StorageClass	cadeia de carateres	sim	Classe de storage a ser usada ao provisionar o volume
tamanho	cadeia de carateres	sim	Tamanho do volume a provisionar em bytes
protocolo	cadeia de carateres	não	Tipo de protocolo a utilizar; "ficheiro" ou "bloco"
InternalName	cadeia de carateres	não	Nome do objeto no sistema de storage; gerado pelo Trident
CloneSourcevolume	cadeia de carateres	não	ONTAP (nas, san) & SolidFire-*: Nome do volume a partir do qual clonar
SplitOnClone	cadeia de carateres	não	ONTAP (nas, san): Divida o clone de seu pai
Política de SnapshotPolicy	cadeia de carateres	não	ONTAP-*: Política de snapshot a ser usada
SnapshotServe	cadeia de carateres	não	ONTAP-*: Porcentagem de volume reservado para snapshots
Política de exportação	cadeia de carateres	não	ONTAP-nas*: Política de exportação para usar

Atributo	Tipo	Obrigatório	Descrição
SnapshotDirectory	bool	não	ONTAP-nas*: Se o diretório snapshot está visível
UnixPermissions	cadeia de carateres	não	ONTAP-nas*: Permissões iniciais do UNIX
Tamanho do bloco	cadeia de carateres	não	SolidFire-*: Tamanho do bloco/setor
Sistema de ficheiros	cadeia de carateres	não	Tipo de sistema de ficheiros

O Trident gera `internalName` ao criar o volume. Isto consiste em duas etapas. Primeiro, ele prepende o prefixo de armazenamento (o padrão `trident` ou o prefixo na configuração de back-end) para o nome do volume, resultando em um nome do formulário `<prefix>-<volume-name>`. Em seguida, procede à higienização do nome, substituindo carateres não permitidos no backend. Para backends ONTAP, ele substitui hífens por sublinhados (assim, o nome interno se torna `<prefix>_<volume-name>`). Para backends de elemento, ele substitui sublinhados por hífens.

Você pode usar configurações de volume para provisionar volumes diretamente usando a API REST, mas nas implantações do Kubernetes, esperamos que a maioria dos usuários use o método padrão do Kubernetes `PersistentVolumeClaim`. O Trident cria esse objeto de volume automaticamente como parte do processo de provisionamento.

ObjetosTrident `Snapshot`

Os snapshots são uma cópia pontual de volumes, que pode ser usada para provisionar novos volumes ou restaurar o estado. No Kubernetes, eles correspondem diretamente a `VolumeSnapshotContent` objetos. Cada snapshot é associado a um volume, que é a origem dos dados do snapshot.

Cada `Snapshot` objeto inclui as propriedades listadas abaixo:

Atributo	Tipo	Obrigatório	Descrição
versão	Cadeia de carateres	Sim	Versão da API Trident ("1")
nome	Cadeia de carateres	Sim	Nome do objeto snapshot Trident
InternalName	Cadeia de carateres	Sim	Nome do objeto snapshot Trident no sistema de storage
Nome do volume	Cadeia de carateres	Sim	Nome do volume persistente para o qual o instantâneo é criado
VolumeInternalName	Cadeia de carateres	Sim	Nome do objeto de volume Trident associado no sistema de storage

No Kubernetes, esses objetos são gerenciados automaticamente. Você pode visualizá-los para ver o que o Trident provisionou.

Quando uma solicitação de objeto Kubernetes `VolumeSnapshot` é criada, o Trident funciona criando um objeto snapshot no sistema de storage de backup. `internalName`O deste objeto instantâneo é gerado combinando o prefixo `snapshot-` com o `UID` do `VolumeSnapshot` objeto (por exemplo, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` e `volumeInternalName` são preenchidos obtendo os detalhes do volume de apoio.

comandos e opções tridentctl

O "Pacote de instalação do Trident" inclui um utilitário de linha de comando `tridentctl` , que fornece acesso simples ao Astra Trident. Usuários do Kubernetes com Privileges suficiente podem usá-lo para instalar o Astra Trident e interagir diretamente com ele para gerenciar o namespace que contém o pod Astra Trident.

Para obter informações de uso, execute `tridentctl --help`o .

Os comandos disponíveis e as opções globais são:

```
Usage:
  tridentctl [command]
```

Comandos disponíveis:

- `create`: Adicionar um recurso ao Astra Trident.
- `delete`: Remova um ou mais recursos do Astra Trident.
- `get`: Obtenha um ou mais recursos do Astra Trident.
- `help`: Ajuda sobre qualquer comando.
- `images`: Imprimir uma tabela das imagens de contêiner que o Astra Trident precisa.
- `import`: Importar um recurso existente para o Astra Trident.
- `install`: Instalar o Astra Trident.
- `logs`: Imprima os logs do Astra Trident.
- `send`: Enviar um recurso de Astra Trident.
- `uninstall`: Desinstalar Astra Trident.
- `update`: Modificar um recurso no Astra Trident.
- `upgrade`: Atualizar um recurso no Astra Trident.
- `version`: Imprimir a versão do Astra Trident.

Bandeiras -

- `-d, --debug`: Saída de depuração.

- `-h, --help`: Ajuda para `tridentctl`.
- `-n, --namespace string`: Namespace da implantação do Astra Trident.
- `-o, --output string`: Formato de saída. Um de JSON|yaml|name|wide|ps (padrão).
- `-s, --server string`: Endereço/porta da interface REST do Astra Trident.

> A interface REST DO Trident pode ser configurada para ouvir e servir apenas em 127.0.0.1 (para IPv4) ou [::1] (para IPv6).

> A interface REST DO Trident pode ser configurada para ouvir e servir apenas em 127.0.0.1 (para IPv4) ou [::1] (para IPv6).

create

Você pode usar executar `create` o comando para adicionar um recurso ao Astra Trident.

```
Usage:
  tridentctl create [option]
```

Opção disponível
`backend`: : Adicionar um back-end ao Astra Trident.

delete

Você pode executar o `delete` comando para remover um ou mais recursos do Astra Trident.

```
Usage:
  tridentctl delete [option]
```

Opções disponíveis:

- `backend`: Excluir um ou mais back-ends de storage do Astra Trident.
- `node`: Excluir um ou mais nós de CSI do Astra Trident.
- `snapshot`: Excluir um ou mais snapshots de volume do Astra Trident.
- `storageclass`: Excluir uma ou mais classes de storage do Astra Trident.
- `volume`: Excluir um ou mais volumes de storage do Astra Trident.

get

Você pode executar o `get` comando para obter um ou mais recursos do Astra Trident.

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

Opções disponíveis:

- `backend`: Obtenha um ou mais back-ends de storage do Astra Trident.
- `snapshot`: Obtenha um ou mais snapshots do Astra Trident.
- `storageclass`: Obtenha uma ou mais classes de storage do Astra Trident.
- `volume`: Obtenha um ou mais volumes do Astra Trident.

images

Você pode executar `images` o sinalizador para imprimir uma tabela das imagens de contentor que o Astra Trident precisa.

```
Usage:
  tridentctl images [flags]
```

Flags: `-h, --help`: Help for images.
* * -v, --k8s-version string': Versão semântica do cluster do Kubernetes.

import volume

Você pode executar o `import volume` comando para importar um volume existente para o Astra Trident.

```
Usage:
  tridentctl import volume <backendName> <volumeName> [flags]
```

Alias:
`volume, v`

Bandeiras -

- `-f, --filename string`: Caminho para o arquivo PVC YAML ou JSON.
- `-h, --help`: Ajuda para volume.
- `--no-manage`: Criar apenas PV/PVC. Não assuma o gerenciamento do ciclo de vida do volume.

install

Você pode executar `install` os sinalizadores para instalar o Astra Trident.

```
Usage:
  tridentctl install [flags]
```

Bandeiras -

- `--autosupport-image string`: A imagem do contentor para telemetria AutoSupport (predefinição "NetApp/Trident AutoSupport:20.07.0").
- `--autosupport-proxy string`: O endereço/porta de um proxy para o envio de telemetria AutoSupport.
- `--csi`: Instalar o CSI Trident (substituir apenas para Kubernetes 1,13, requer portões de recurso).
- `--enable-node-prep`: Tentativa de instalar os pacotes necessários nos nós.
- `--generate-custom-yaml`: Gere arquivos YAML sem instalar nada.
- `-h, --help`: Ajuda para instalar.
- `--http-request-timeout`: Substituir o tempo limite da solicitação HTTP para a API REST do controlador Trident (1m30s padrão).
- `--image-registry string`: O endereço/porta de um Registro de imagem interno.
- `--k8s-timeout duration`: O tempo limite para todas as operações do Kubernetes (3m0s padrão).
- `--kubelet-dir string`: A localização do host do estado interno do kubelet (padrão "/var/lib/kubelet").
- `--log-format string`: O formato de log Astra Trident (texto, json) (texto padrão).
- `--pv string`: O nome do PV legado usado pelo Astra Trident garante que isso não existe (padrão "Trident").
- `--pvc string`: O nome do PVC legado usado pelo Astra Trident garante que isso não existe (padrão "Trident").
- `--silence-autosupport`: Não envie pacotes AutoSupport automaticamente para o NetApp (padrão verdadeiro).
- `--silent`: Desativar a saída MOST durante a instalação.
- `--trident-image string`: A imagem Astra Trident a instalar.
- `--use-custom-yaml`: Use todos os arquivos YAML existentes que existem no diretório de configuração.
- `--use-ipv6`: Utilizar o IPv6 para a comunicação do Astra Trident.

`logs`

Você pode executar `logs` os sinalizadores para imprimir os logs do Astra Trident.

```
Usage:
  tridentctl logs [flags]
```

Bandeiras -

- `-a, --archive`: Crie um arquivo de suporte com todos os logs, a menos que especificado de outra forma.
- `-h, --help`: Ajuda para logs.
- `-l, --log string`: Log do Astra Trident para exibição. Um dos Trident|auto|Trident-operator|All (predefinição "auto").
- `--node string`: O nome do nó Kubernetes do qual você pode coletar logs do pod de nó.
- `-p, --previous`: Obter os logs para a instância de contentor anterior, se ela existir.
- `--sidecars`: Obter os logs para os recipientes sidecar.

send

Você pode executar o `send` comando para enviar um recurso do Astra Trident.

```
Usage:
  tridentctl send [option]
```

Opção disponível
`autosupport`: : Enviar um arquivo AutoSupport para o NetApp.

uninstall

Você pode executar `uninstall` os sinalizadores para desinstalar o Astra Trident.

```
Usage:
  tridentctl uninstall [flags]
```

Bandeiras: * `-h, --help`: Ajuda para desinstalar. `--silent`* : Desative a saída MOST durante a desinstalação.

update

Você pode executar os `update` comandos para modificar um recurso no Astra Trident.

```
Usage:
  tridentctl update [option]
```

Opções disponíveis
`backend`: : Atualize um back-end no Astra Trident.

upgrade

Você pode executar os `upgrade` comandos para atualizar um recurso no Astra Trident.

```
Usage:
tridentctl upgrade [option]
```

Opção disponível
`volume`: : Atualize um ou mais volumes persistentes de NFS/iSCSI para CSI.

`version`

Você pode executar `version` os sinalizadores para imprimir a versão do `tridentctl` e o serviço Trident em execução.

```
Usage:
  tridentctl version [flags]
```

Bandeiras: * `--client`: Apenas versão do cliente (nenhum servidor necessário). `-h, --help`* : Ajuda para a versão.

Versões anteriores da documentação

Se você não estiver executando o Astra Trident 22,04, a documentação das versões anteriores estará disponível.

- "Astra Trident 22,01"

Avisos legais

Avisos legais fornecem acesso a declarações de direitos autorais, marcas registradas, patentes e muito mais.

Direitos de autor

"https://www.netapp.com/company/legal/copyright/"

Marcas comerciais

NetApp, o logotipo DA NetApp e as marcas listadas na página de marcas comerciais da NetApp são marcas comerciais da NetApp, Inc. Outros nomes de produtos e empresas podem ser marcas comerciais de seus respetivos proprietários.

"https://www.netapp.com/company/legal/trademarks/"

Patentes

Uma lista atual de patentes de propriedade da NetApp pode ser encontrada em:

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Política de privacidade

"https://www.netapp.com/company/legal/privacy-policy/"

Código aberto

Você pode revisar os direitos autorais e as licenças de terceiros usadas no software NetApp para Astra Trident no arquivo de avisos de cada versão em https://github.com/NetApp/trident/.

Informações sobre direitos autorais

Copyright © 2024 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 DESTE DOCUMENTO. 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 SOBRESSALENTES; 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 DESTE SOFTWARE, 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.