Referência

Trident

NetApp
July 01, 2026

Índice

Referência

Portas Trident

Saiba mais sobre as portas que Trident utiliza para comunicação.

Visão geral

Trident utiliza diversas portas para comunicação dentro de clusters Kubernetes e com storage backends. A seguir, um resumo das principais portas, suas finalidades e considerações de segurança.

- **Foco em tráfego de saída**: Os nós do Kubernetes (controller e worker) iniciam principalmente o tráfego para LIFs/IPs de storage, portanto, as regras do iptables devem permitir tráfego de saída dos IPs dos nós para IPs de storage específicos nessas portas. Evite regras amplas do tipo "qualquer para qualquer".
- **Restrições de entrada**: limite as portas internas do Trident ao tráfego interno do cluster (por exemplo, usando CNI como Calico). Nenhuma exposição desnecessária de entrada nos firewalls do host.
- **Segurança de protocolo**:
  - Use TCP sempre que possível (mais confiável).
  - Habilite CHAP/IPsec para iSCSI se sensível; TLS/HTTPS para gerenciamento (porta 443/8443).
  - Para NFSv4 (padrão no Trident), remova as portas UDP/NFSv3 mais antigas (por exemplo, 4045-4049) se não forem necessárias.
  - Restrinja o acesso a sub-redes confiáveis; monitore com ferramentas como Prometheus (porta 8001 opcional).

Portas para nós controladores

Essas portas são destinadas principalmente ao Trident operator (gerenciamento de backend). Todas as portas internas são de nível de pod; permita nos nós somente se o firewall do host interferir com o CNI.

Porta/Protocolo	Direção	Propósito	Driver/Protocolo	Notas de segurança
TCP 8000	Entrada/Saída (cluster-internal)	Servidor REST Trident (comunicação operador-controlador)	Todos	Restringir aos CIDRs dos pods; sem exposição externa.
TCP 8443	Entrada/Saída (cluster-internal)	Backchannel HTTPS (API interna segura)	Todos	Criptografia TLS; limite ao service mesh do Kubernetes se utilizado.
TCP 8001	Entrada (interna ao cluster, opcional)	Métricas Prometheus	Todos	Exponha apenas às ferramentas de monitoramento (por exemplo, usando RBAC); desative se não for usado.
TCP 443	Saída	HTTPS para ONTAP SVM/cluster mgmt LIF	ONTAP (todos), ANF	Exigir validação de certificado TLS; restringir apenas aos IPs de mgmt do LIF.

Porta/Protocolo	Direção	Propósito	Driver/Protocolo	Notas de segurança
TCP 8443	Saída	HTTPS para Proxy de Serviços Web E-Series	E-Series (iSCSI)	API REST padrão; utiliza certificados; configurável no YAML do backend.

Portas para nós de trabalho

Essas portas são para daemonsets de nós CSI e montagens de pods. As portas de dados são de saída para LIFs de dados de storage; inclua extras do NFSv3 se estiver usando NFSv3 (opcional para NFSv4).

Porta/Protocolo	Direção	Propósito	Driver/Protocolo	Notas de segurança
TCP 17546	Entrada (local para pod)	Sondas de liveness/prontidão do nó CSI	Todos	Configurável (--probe-port); garanta que não haja conflitos de host; somente local.
TCP 8000	Entrada/Saída (cluster-internal)	Servidor REST Trident	Todos	Como acima; interno ao pod.
TCP 8443	Entrada/Saída (cluster-internal)	HTTPS de canal de retorno	Todos	Como acima.
TCP 8001	Entrada (interna ao cluster, opcional)	Métricas Prometheus	Todos	Como acima.
TCP 443	Saída	HTTPS para ONTAP SVM/cluster mgmt LIF	ONTAP (todos), ANF	Conforme mencionado acima; usado para descoberta.
TCP 8443	Saída	HTTPS para Proxy de Serviços Web E-Series	E-Series (iSCSI)	Como acima.
TCP/UDP 111	Saída	RPCBIND/portmapper	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Obrigatório para v3; opcional para v4 (descarregamento do firewall); restringir se estiver usando somente NFSv4.
TCP/UDP 2049	Saída	Daemon NFS	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Dados essenciais; bem conhecidos; use TCP para confiabilidade.
TCP/UDP 635	Saída	Daemon de montagem	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Montagem; callbacks bidirecionais possíveis (permitir entrada efêmera se necessário).
UDP 4045	Saída	Gerenciador de bloqueio NFS (nlockmgr)	ONTAP-NAS (NFSv3)	Bloqueio de arquivos; ignorar para v4 (pNFS handles); somente UDP.

Porta/Protocolo	Direção	Propósito	Driver/Protocolo	Notas de segurança
UDP 4046	Saída	Monitor de status NFS (statd)	ONTAP-NAS (NFSv3)	Notificações; podem ser necessárias portas efêmeras de entrada (1024-65535) para callbacks.
UDP 4049	Saída	Daemon de cotas NFS (rquotad)	ONTAP-NAS (NFSv3)	Cotas; pular para v4.
TCP 3260	Saída	destino iSCSI (descoberta/dados/CHAP)	ONTAP-SAN (iSCSI), E-Series (iSCSI)	Bem conhecido; autenticação CHAP nesta porta; habilite CHAP mútuo para segurança.
TCP 445	Saída	SMB/CIFS	ONTAP-NAS (SMB), ANF (SMB)	Bem conhecido; use SMB3 com criptografia (Trident annotation netapp.io/smb-encryption=true).
TCP/UDP 88 (opcional)	Saída	Autenticação Kerberos	ONTAP (NFS/SMB/iSCSI com Kerberos)	Se estiver usando Kerberos (não é o padrão); para servidores AD, não para storage.
TCP/UDP 389 (opcional)	Saída	LDAP	ONTAP (NFS/SMB com LDAP)	Semelhante; para resolução de nomes/autenticação; restringir ao AD.

A porta de liveness/readiness pode ser alterada durante a instalação usando a flag `--probe-port`. É importante garantir que essa porta não esteja sendo usada por outro processo nos nós de trabalho.

API REST Trident

Embora "Comandos e opções do tridentctl" sejam a maneira mais fácil de interagir com a Trident API REST, você pode usar o endpoint REST diretamente se preferir.

Quando usar a API REST

API REST é útil para instalações avançadas que usam Trident como um binário independente em implantações não-Kubernetes.

Para maior segurança, o Trident `REST API` é restrito ao localhost por padrão quando executado dentro de um pod. Para alterar esse comportamento, você precisa definir o argumento do Trident `-address` em sua configuração de pod.

Usando API REST

Para exemplos de como essas APIs são chamadas, passe a flag (`-d` de debug. Para obter mais informações, consulte "Gerencie Trident usando tridentctl".

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 "Gerencie Trident usando tridentctl".
- Se o objeto já existir, o comportamento varia: backends atualizam o objeto existente, enquanto todos os outros tipos de objeto falharão a operação.

EXCLUIR

`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 "Gerencie Trident usando tridentctl".

Opções de linha de comando

Trident expõe diversas opções de linha de comando para o orquestrador Trident. Você pode usar essas opções para modificar sua implantação.

Log

`-debug`

Ativa a saída de depuração.

`-loglevel <level>`

Define o nível de registro (debug, info, warn, error, fatal). O padrão é info.

Kubernetes

`-k8s_pod`

Use esta opção ou `-k8s_api_server` para habilitar o suporte ao Kubernetes. Ao configurar esta opção, Trident utiliza as credenciais da conta de serviço do Kubernetes do pod que o contém para contatar o servidor da API. Isso só funciona quando Trident é executado como um pod em um cluster Kubernetes com contas de serviço habilitadas.

-k8s_api_server <insecure-address:insecure-port>

Use esta opção ou `-k8s_pod` para habilitar o suporte ao Kubernetes. Quando especificado, Trident se conecta ao servidor da 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 suporta apenas conexões inseguras com o servidor da API. Para se conectar com segurança, implante Trident em um pod com a opção `-k8s_pod`.

Docker

-volume_driver <name>

Nome do driver usado ao registrar o plugin do Docker. O padrão é `netapp`.

-driver_port <port-number>

Escute nesta porta em vez de um socket de domínio UNIX.

-config <file>

Obrigatório; você deve especificar este caminho para um arquivo de configuração de backend.

REST

-address <ip-or-host>

Especifica o endereço no qual o servidor REST do Trident deve escutar. O padrão é localhost. Quando escuta em localhost e está em execução dentro de um pod do Kubernetes, a interface REST não é diretamente acessível de fora do pod. Use `-address ""` para tornar a interface REST acessível a partir do endereço IP do pod.

A interface REST do Trident pode ser configurada para escutar 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

Habilita a interface REST. O valor padrão é true.

Objetos Kubernetes e Trident

Você pode interagir com Kubernetes e Trident usando APIs REST, lendo e gravando objetos de recursos. Existem diversos objetos de recursos que definem a relação entre Kubernetes e Trident, Trident e storage, e Kubernetes e storage. Alguns desses objetos são gerenciados por meio do Kubernetes e outros são gerenciados por meio do Trident.

Como os objetos interagem entre si?

Talvez a maneira mais fácil de entender os objetos, para que servem e como interagem, seja acompanhar uma única solicitação de storage de um usuário do Kubernetes:

1. Um usuário cria uma `PersistentVolumeClaim` solicitando um novo `PersistentVolume` de um tamanho específico de um `StorageClass` Kubernetes que foi previamente configurado pelo administrador.

2. O Kubernetes `StorageClass` identifica Trident como seu provisionador e inclui parâmetros que informam Trident como provisionar um volume para a classe solicitada.
3. Trident analisa seus próprios `StorageClass` com o mesmo nome que identifica a correspondência `Backends` e `StoragePools` que pode usar para provisionar volumes para a classe.
4. Trident provisiona storage em um backend correspondente e cria dois objetos: um `PersistentVolume` no Kubernetes que informa 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 o `PersistentVolumeClaim` ao novo `PersistentVolume`. Os pods que incluem o `PersistentVolumeClaim` montam esse PersistentVolume em qualquer host em que sejam executados.
6. Um usuário cria um `VolumeSnapshot` de um PVC existente, usando um `VolumeSnapshotClass` que aponta para Trident.
7. Trident identifica o volume associado ao PVC e cria um snapshot do volume em seu backend. Ele também cria um `VolumeSnapshotContent` que instrui o Kubernetes sobre como identificar o snapshot.
8. Um usuário pode criar um `PersistentVolumeClaim` usando `VolumeSnapshot` como a fonte.
9. Trident identifica o snapshot necessário e executa o mesmo conjunto de etapas envolvidas na criação de `PersistentVolume` e de `Volume`.

Para obter mais informações sobre objetos do Kubernetes, recomendamos fortemente que você leia a "Volumes persistentes" seção da documentação do Kubernetes.

Kubernetes `PersistentVolumeClaim` objetos

Um objeto `PersistentVolumeClaim`Kubernetes é uma solicitação de armazenamento feita por um usuário do cluster Kubernetes.

Além da especificação padrão, Trident permite que os usuários especifiquem as seguintes anotações específicas de volume se quiserem substituir os valores padrão que você definiu na configuração do backend:

Anotação	Opção de volume	Drivers com suporte
trident.netapp.io/fileSystem	fileSystem	ontap-san, solidfire-san,ontap-san-economy
trident.netapp.io/cloneFromPVC	cloneSourceVolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ontap-nas, ontap-san
trident.netapp.io/protocol	protocolo	qualquer
trident.netapp.io/exportPolicy	exportPolicy	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	snapshotPolicy	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotReserve	snapshotReserve	ontap-nas, ontap-nas-flexgroup, ontap-san

Anotação	Opção de volume	Drivers com suporte
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	blockSize	solidfire-san
trident.netapp.io/skipRecoveryQueue	skipRecoveryQueue	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy

Se o PV criado tiver a `Delete` política de recuperação, Trident exclui tanto o PV quanto o volume de suporte quando o PV é liberado (ou seja, quando o usuário exclui o PVC). Caso a ação de exclusão falhe, Trident marca o PV como tal e tenta a operação periodicamente até que seja bem-sucedida ou o PV seja excluído manualmente. Se o PV usar a `Retain` política, Trident a ignora e assume que o administrador o removerá do Kubernetes e do backend, permitindo que o volume seja copiado ou inspecionado antes de sua remoção. Observe que a exclusão do PV não faz com que Trident exclua o volume de suporte. Você deve removê-lo usando a API REST (`tridentctl`.

Trident suporta a criação de Snapshots de Volume usando a especificação CSI: você pode criar um Snapshot de Volume e usá-lo como uma fonte de dados para clonar PVCs existentes. Dessa forma, cópias ponto no tempo de PVs podem ser expostas ao Kubernetes na forma de snapshots. Os snapshots podem então ser usados para criar novos PVs. Veja `On-Demand Volume Snapshots` para ver como isso funcionaria.

Trident também fornece as `cloneFromPVC` e `splitOnClone`anotações para criar clones. Você pode usar essas anotações para clonar um PVC sem precisar usar a implementação CSI.

Aqui está um exemplo: se um usuário já possui um PVC chamado `mysql`, o usuário pode criar um novo PVC chamado `mysqlclone` usando a anotação, como `trident.netapp.io/cloneFromPVC: mysql`. Com essa anotação definida, Trident clona o volume correspondente ao PVC mysql, em vez de provisionar um volume do zero.

Considere os seguintes pontos:

- NetApp recomenda clonar um volume ocioso.
- Um PVC e seu clone devem estar no mesmo namespace do Kubernetes e ter a mesma storage class.
- Com os `ontap-nas` e `ontap-san` drivers, pode ser desejável definir a anotação PVC `trident.netapp.io/splitOnClone` em conjunto com `trident.netapp.io/cloneFromPVC`. Com `trident.netapp.io/splitOnClone` definido como `true`, o Trident divide o volume clonado do volume pai, desacoplando completamente o ciclo de vida do volume clonado do seu volume pai, à custa de alguma perda de eficiência de storage. Não definir `trident.netapp.io/splitOnClone` ou defini-lo como `false` resulta em menor consumo de espaço no backend, à custa de criar dependências entre os volumes pai e clone, de forma 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 se espera que o volume e seu clone divirjam bastante e não se beneficiem das eficiências de storage oferecidas pelo ONTAP.

O `sample-input` diretório contém exemplos de definições de PVC para uso com Trident. Consulte para obter uma descrição completa dos parâmetros e configurações associados aos volumes do Trident.

Kubernetes `PersistentVolume` objetos

Um objeto Kubernetes `PersistentVolume` representa um espaço de armazenamento disponibilizado para o cluster Kubernetes. Ele possui um ciclo de vida independente do pod que o utiliza.

> Trident cria `PersistentVolume` objetos e os registra automaticamente no cluster Kubernetes com base nos volumes que ele provisiona. Não é necessário que você os gerencie.

Ao criar um PVC que se refere a um Trident-based `StorageClass`, 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, Trident segue as seguintes regras:

- Trident gera um nome de PV para Kubernetes e um nome interno que utiliza para provisionar o storage. Em ambos os casos, garante que os nomes sejam únicos em seu escopo.
- O tamanho do volume corresponde o mais próximo possível ao tamanho solicitado no PVC, embora possa ser arredondado para a quantidade alocável mais próxima, dependendo da plataforma.

Kubernetes `StorageClass` objetos

Objetos do Kubernetes `StorageClass` são especificados por nome em `PersistentVolumeClaims` para provisionar armazenamento com um conjunto de propriedades. A própria storage class 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 de backend do Trident.

Um objeto Kubernetes `StorageClass` que usa Trident tem a seguinte aparência:

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

Esses parâmetros são específicos do Trident e informam ao Trident como provisionar volumes para a classe.

Os parâmetros da classe de armazenamento são:

Atributo	Tipo	Obrigatório	Descrição
atributos	map[string]string	não	Consulte a seção de atributos abaixo
storagePools	map[string]StringList	não	Mapeamento de nomes de backend para listas de pools de storage dentro

Atributo	Tipo	Obrigatório	Descrição
additionalStoragePools	map[string]StringList	não	Mapeamento de nomes de backend para listas de pools de storage dentro
excludeStoragePools	map[string]StringList	não	Mapeamento de nomes de backend para listas de pools de storage dentro

Os atributos de armazenamento e seus possíveis valores podem ser classificados em atributos de seleção de pool de storage e atributos do Kubernetes.

Atributos de seleção do storage pool

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

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

Atributo	Tipo	Valores	Oferta	Solicitação	Apoiado por
IOPS	inteiro	inteiro positivo	O pool é capaz de garantir IOPS nessa faixa	Volume garantiu esses IOPS	solidfire-san

[1]: não suportado pelos sistemas ONTAP Select

Na maioria dos casos, os valores solicitados influenciam diretamente o provisionamento; por exemplo, solicitar provisionamento espesso resulta em um volume provisionado de forma espessa. No entanto, um pool de storage usa seus valores mínimo e máximo de IOPS oferecidos para definir os valores de QoS, em vez do valor solicitado. Nesse caso, o valor solicitado é usado apenas para selecionar o pool de storage.

Idealmente, você pode usar `attributes` sozinho para modelar as qualidades do armazenamento necessário para atender às necessidades de uma classe específica. Trident descobre e seleciona automaticamente pools de storage que correspondem a *todos* os `attributes` que você especificar.

Caso você não consiga usar `attributes` para selecionar automaticamente os pools corretos para uma classe, você pode usar os parâmetros `storagePools` e `additionalStoragePools` para refinar ainda mais os pools ou até mesmo para 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, Trident usa a interseção dos pools identificados pelos parâmetros `attributes` e `storagePools` para provisionamento. Você pode usar qualquer um dos parâmetros sozinho ou ambos juntos.

Você pode usar o `additionalStoragePools` parâmetro para estender o conjunto de pools que Trident usa para provisionamento, independentemente de quaisquer pools selecionados pelos `attributes` e `storagePools` parâmetros.

Você pode usar o `excludeStoragePools`parâmetro para filtrar o conjunto de pools que Trident usa para provisionamento. Usar esse parâmetro remove quaisquer pools correspondentes.

Nos `storagePools` e `additionalStoragePools` parâmetros, cada entrada assume o formato `<backend>:<storagePoolList>`, onde `<storagePoolList>` é uma lista de pools de storage separados por vírgula para o backend especificado. Por exemplo, um valor para `additionalStoragePools` pode ser algo como `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`. Essas listas aceitam valores regex tanto para o backend quanto para os valores da lista. Você pode usar `tridentctl get backend` para obter a lista de backends e seus pools.

Atributos do Kubernetes

Esses atributos não têm impacto na seleção de pools/backends de storage pelo Trident durante o provisionamento dinâmico. Em vez disso, esses atributos simplesmente fornecem parâmetros suportados pelos Volumes Persistentes do Kubernetes. Os nós de trabalho são responsáveis pelas operações de criação do sistema de arquivos e podem exigir utilitários de sistema de arquivos, como xfsprogs.

Atributo	Tipo	Valores	Descrição	Drivers relevantes	Versão do Kubernetes
fsType	string	ext4, ext3, xfs	O tipo de sistema de arquivos para volumes em bloco	solidfire-san, ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy	Todos
allowVolumeExpansion	booleano	true, false	Ativar ou desativar o suporte para aumentar o tamanho do PVC	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, azure-netapp-files	1.11+
volumeBindingMode	string	Imediato, WaitForFirstConsumer	Escolha quando a vinculação de volume e o provisionamento dinâmico ocorre	Todos	1.19 - 1.26

- 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 storage class para indicar que existe um sistema de arquivos. A propriedade do volume pode ser controlada usando o `fsGroup` security context de um pod somente se `fsType` estiver definido. Consulte "Kubernetes: configurar um contexto de segurança para um Pod ou contêiner" para uma visão geral sobre como definir a propriedade do volume usando o `fsGroup` context. O Kubernetes aplicará o valor `fsGroup` somente se:
  - `fsType` está definido na classe de armazenamento.
  - O modo de acesso do PVC é RWO.

  Para drivers de armazenamento NFS, um sistema de arquivos já existe como parte da exportação NFS. Para usar `fsGroup` a classe de armazenamento ainda é necessário especificar um `fsType`. Você pode defini-lo como `nfs` ou qualquer valor não nulo.
- Consulte "Expandir volumes" para obter mais detalhes sobre expansão de volume.
- O pacote de instalação do Trident fornece vários exemplos de definições de classes de armazenamento para uso com Trident `sample-input/storage-class-*.yaml`. A exclusão de uma classe de armazenamento do Kubernetes faz com que a classe de armazenamento correspondente do Trident também seja excluída.

Kubernetes `VolumeSnapshotClass` objetos

Objetos do Kubernetes `VolumeSnapshotClass` são análogos a `StorageClasses`. Eles ajudam a definir

várias classes de storage e são referenciados por snapshots de volume para associar o snapshot à classe de snapshot necessária. Cada snapshot de volume está associado a uma única classe de snapshot de volume.

A `VolumeSnapshotClass` deve ser definida por um administrador para criar snapshots. Uma classe de snapshot de volume é criada com a seguinte definição:

```
apiVersion: snapshot.storage.k8s.io/v1
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 da `csi-snapclass` classe são tratadas pelo Trident. O `deletionPolicy` especifica a ação a ser tomada quando um snapshot precisa ser excluído. Quando `deletionPolicy` é definido como `Delete`, os objetos de snapshot de volume, bem como o snapshot subjacente no cluster de storage, são removidos quando um snapshot é excluído. Alternativamente, defini-lo como `Retain` significa que `VolumeSnapshotContent` e o snapshot físico são mantidos.

Kubernetes `VolumeSnapshot` objetos

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 snapshot de volume é uma solicitação feita por um usuário para criar um snapshot de um PVC existente.

Quando uma solicitação de snapshot de volume é recebida, Trident gerencia automaticamente a criação do snapshot para o volume no backend e expõe o snapshot criando um objeto `VolumeSnapshotContent` exclusivo. Você pode criar snapshots a partir de PVCs existentes e usar os snapshots como DataSource ao criar novos PVCs.

> O ciclo de vida de um VolumeSnapshot é independente do PVC de origem: um snapshot persiste mesmo após a exclusão do PVC de origem. Ao excluir um PVC que possui snapshots associados, Trident marca o volume de suporte desse PVC no estado **Deleting**, mas não o remove completamente. O volume é removido quando todos os snapshots associados são excluídos.

Kubernetes `VolumeSnapshotContent` objetos

Um objeto `VolumeSnapshotContent`Kubernetes representa um snapshot obtido de um volume já provisionado. É análogo a um `PersistentVolume` e indica um snapshot provisionado no cluster de storage. Semelhante a `PersistentVolumeClaim` e `PersistentVolume` objetos, quando um snapshot é criado, o objeto `VolumeSnapshotContent` mantém um mapeamento um-para-um com o objeto `VolumeSnapshot` que solicitou a criação do snapshot.

O `VolumeSnapshotContent` objeto contém detalhes que identificam exclusivamente o snapshot, como o `snapshotHandle`. Esta `snapshotHandle` é uma combinação única do nome do PV e do nome do `VolumeSnapshotContent` objeto.

Quando uma solicitação de snapshot é recebida, Trident cria o snapshot no backend. Após a criação do

snapshot, Trident configura um `VolumeSnapshotContent` objeto e assim expõe o snapshot à API do Kubernetes.

> Normalmente, você não precisa gerenciar o `VolumeSnapshotContent` objeto. Uma exceção a isso é quando você deseja "importar um snapshot de volume" criado fora do Trident.

Kubernetes `VolumeGroupSnapshotClass` objetos

Objetos do Kubernetes `VolumeGroupSnapshotClass` são análogos a `VolumeSnapshotClass`. Eles ajudam a definir várias classes de storage e são referenciados por snapshots de grupo de volume para associar o snapshot à classe de snapshot necessária. Cada snapshot de grupo de volume está associado a uma única classe de snapshot de grupo de volume.

A `VolumeGroupSnapshotClass` deve ser definida por um administrador para criar um grupo de snapshots. Uma classe de snapshot de grupo de volumes é criada com a seguinte definição:

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

O `driver` especifica ao Kubernetes que as solicitações para snapshots de grupo de volumes da classe `csi-group-snap-class` são tratadas pelo Trident. O `deletionPolicy` especifica a ação a ser tomada quando um snapshot de grupo precisa ser excluído. Quando `deletionPolicy` é definido como `Delete`, os objetos de snapshot do grupo de volumes, bem como o snapshot subjacente no cluster de storage, são removidos quando um snapshot é excluído. Alternativamente, defini-lo como `Retain` significa que `VolumeGroupSnapshotContent` e o snapshot físico são mantidos.

Kubernetes `VolumeGroupSnapshot` objetos

Um objeto do Kubernetes `VolumeGroupSnapshot` é uma solicitação para criar um snapshot de múltiplos volumes. Assim como um PVC representa uma solicitação feita por um usuário para um volume, um snapshot de grupo de volumes é uma solicitação feita por um usuário para criar um snapshot de um PVC existente.

Quando uma solicitação de snapshot de grupo de volumes é recebida, Trident gerencia automaticamente a criação do snapshot do grupo para os volumes no backend e expõe o snapshot criando um objeto `VolumeGroupSnapshotContent` exclusivo. Você pode criar snapshots a partir de PVCs existentes e usar os snapshots como DataSource ao criar novos PVCs.

> O ciclo de vida de um VolumeGroupSnapshot é independente do PVC de origem: um snapshot persiste mesmo após a exclusão do PVC de origem. Ao excluir um PVC que possui snapshots associados, Trident marca o volume de suporte desse PVC no estado **Deleting**, mas não o remove completamente. O snapshot do grupo de volumes é removido quando todos os snapshots associados são excluídos.

Kubernetes `VolumeGroupSnapshotContent` objetos

Um objeto Kubernetes `VolumeGroupSnapshotContent` representa um snapshot de grupo obtido de um volume já provisionado. É análogo a um `PersistentVolume` e indica um snapshot provisionado no cluster de storage. Semelhante a `PersistentVolumeClaim` e `PersistentVolume` objetos, quando um snapshot é criado, o objeto `VolumeSnapshotContent` mantém um mapeamento um-para-um com o objeto `VolumeSnapshot` que solicitou a criação do snapshot.

O `VolumeGroupSnapshotContent` objeto contém detalhes que identificam o grupo de snapshots, como o `volumeGroupSnapshotHandle` e os volumeSnapshotHandles individuais existentes no sistema de storage.

Quando uma solicitação de snapshot é recebida, Trident cria o snapshot do grupo de volume no backend. Após a criação do snapshot do grupo de volume, Trident configura um `VolumeGroupSnapshotContent` objeto e, assim, expõe o snapshot à API do Kubernetes.

Kubernetes `CustomResourceDefinition` objetos

Os recursos personalizados do Kubernetes são endpoints na API do Kubernetes definidos pelo administrador e usados para agrupar objetos semelhantes. O Kubernetes oferece suporte à criação de recursos personalizados para armazenar uma coleção de objetos. Você pode obter essas definições de recursos executando `kubectl get crds`.

As definições de recursos personalizados (CRDs) e seus metadados de objeto são armazenados pelo Kubernetes em seu repositório de metadados. Isso elimina a necessidade de um repositório separado para Trident.

Trident usa `CustomResourceDefinition` objetos para preservar a identidade de objetos Trident, como backends Trident, classes de storage Trident e volumes Trident. Esses objetos são gerenciados pelo Trident. Além disso, a estrutura de snapshot de volume CSI introduz alguns CRDs que são necessários para definir snapshots de volume.

Os CRDs são uma construção do Kubernetes. Objetos dos recursos definidos acima são criados pelo Trident. Como um exemplo simples, quando um backend é criado usando `tridentctl`, um objeto CRD correspondente `tridentbackends` é criado para consumo pelo Kubernetes.

Aqui estão alguns pontos a serem lembrados sobre os CRDs do Trident:

- Quando Trident é instalado, um conjunto de CRDs é criado e pode ser usado como qualquer outro tipo de recurso.
- Ao desinstalar Trident usando o comando `tridentctl uninstall`, os pods do Trident são excluídos, mas os CRDs criados não são removidos. Consulte "Desinstalar Trident" para entender como Trident pode ser completamente removido e reconfigurado do zero.

Trident `StorageClass` objetos

Trident cria classes de armazenamento correspondentes para objetos do Kubernetes `StorageClass` que especificam `csi.trident.netapp.io` em seu campo provisioner. O nome da classe de armazenamento corresponde ao do objeto do Kubernetes `StorageClass` que ela representa.

> Com o Kubernetes, esses objetos são criados automaticamente quando um Kubernetes `StorageClass` que usa Trident como provisionador é registrado.

As classes de armazenamento compreendem um conjunto de requisitos para volumes. Trident corresponde esses requisitos com os atributos presentes em cada pool de storage; se eles corresponderem, esse pool de storage é um destino válido para o provisionamento de volumes usando essa classe de armazenamento.

Você pode criar configurações de classes de armazenamento para definir diretamente as classes de armazenamento usando a API REST. No entanto, para implantações do Kubernetes, esperamos que elas sejam criadas ao registrar novos objetos `StorageClass` do Kubernetes.

Objetos de backend Trident

Os backends representam os provedores de armazenamento sobre os quais Trident provisiona volumes; uma única instância do Trident pode gerenciar qualquer número de backends.

Este é um dos dois tipos de objeto que você cria e gerencia sozinho. O outro é o objeto `StorageClass`Kubernetes.

Para obter mais informações sobre como construir esses objetos, consulte "configurando backends".

Trident `StoragePool` objetos

Os pools de storage representam os locais distintos disponíveis para provisionamento em cada backend. Para ONTAP, eles correspondem a agregados em SVMs. Para NetApp HCI/SolidFire, eles correspondem a bandas de QoS especificadas pelo administrador. Cada pool de storage possui um conjunto de atributos de storage distintos, que definem suas características de desempenho e de proteção de dados.

Diferentemente dos outros objetos aqui, os candidatos a pool de storage são sempre descobertos e gerenciados automaticamente.

Trident `Volume` objetos

Volumes são a unidade básica de provisionamento, compreendendo endpoints de backend, como compartilhamentos NFS, e LUNs iSCSI e FC. No Kubernetes, estes correspondem diretamente a `PersistentVolumes`. 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 Trident provisionou.
- Ao excluir um PV com snapshots associados, o volume Trident correspondente é atualizado para o estado **Deleting**. 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	string	não	Versão da API Trident ("1")
nome	string	sim	Nome do volume a ser criado

Atributo	Tipo	Obrigatório	Descrição
storageClass	string	sim	Classe de storage a ser usada ao provisionar o volume
tamanho	string	sim	Tamanho do volume a ser provisionado em bytes
protocolo	string	não	Tipo de protocolo a ser usado; "arquivo" ou "bloco"
internalName	string	não	Nome do objeto no sistema de storage; gerado pelo Trident
cloneSourceVolume	string	não	ontap (nas, san) e solidfire-*: Nome do volume do qual clonar
splitOnClone	string	não	ontap (nas, san): separar o clone de seu progenitor
snapshotPolicy	string	não	ontap-*: política do Snapshot a ser usada
snapshotReserve	string	não	ontap-*: Percentual do volume reservado para snapshots
exportPolicy	string	não	ontap-nas*: política de exportação a ser usada
snapshotDirectory	bool	não	ontap-nas*: se o diretório de snapshot está visível
unixPermissions	string	não	ontap-nas*: Permissões iniciais do UNIX
blockSize	string	não	solidfire-*: Tamanho do bloco/setor
fileSystem	string	não	Tipo de sistema de arquivos
skipRecoveryQueue	string	não	Durante a exclusão de volume, ignore a fila de recuperação no armazenamento e exclua o volume imediatamente.

Trident gera `internalName` ao criar o volume. Isso consiste em duas etapas. Primeiro, ele adiciona o prefixo de storage (o padrão `trident` ou o prefixo na configuração do backend) ao nome do volume, resultando em um nome no formato `<prefix>-<volume-name>`. Em seguida, ele sanitiza o nome, substituindo caracteres não permitidos no backend. Para backends ONTAP, ele substitui hífens por sublinhados (assim, o nome interno se torna `<prefix>_<volume-name>`). Para backends Element, ele substitui sublinhados por hífens.

Você pode usar configurações de volume para provisionar volumes diretamente usando a API REST, mas em

implantações do Kubernetes esperamos que a maioria dos usuários utilize o método padrão do Kubernetes `PersistentVolumeClaim`. Trident cria esse objeto de volume automaticamente como parte do processo de provisionamento.

Trident `Snapshot` objetos

Snapshots são uma cópia pontual de volumes, que podem ser usadas para provisionar novos volumes ou restaurar o estado. No Kubernetes, estes correspondem diretamente a `VolumeSnapshotContent` objetos. Cada snapshot está associado a um volume, que é a fonte dos dados para o snapshot.

Cada objeto `Snapshot` inclui as propriedades listadas abaixo:

Atributo	Tipo	Obrigatório	Descrição
versão	String	Sim	Versão da API Trident ("1")
nome	String	Sim	Nome do objeto de Snapshot do Trident
internalName	String	Sim	Nome do objeto de snapshot do Trident no sistema de storage
volumeName	String	Sim	Nome do Volume Persistente para o qual o snapshot é criado
volumeInternalName	String	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 Trident provisionou.

Quando uma solicitação de objeto Kubernetes `VolumeSnapshot` é criada, Trident funciona criando um objeto de snapshot no sistema de storage. O `internalName` desse objeto de snapshot 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 subjacente.

Trident `ResourceQuota` objeto

O Trident daemonset consome uma `system-node-critical` Priority Class—a Priority Class mais alta disponível no Kubernetes—para garantir que Trident possa identificar e limpar volumes durante o desligamento correto do nó e permitir que os pods do Trident daemonset preemptem cargas de trabalho com prioridade mais baixa em clusters com alta pressão de recursos.

Para isso, Trident utiliza um `ResourceQuota` objeto para garantir que a Classe de Prioridade "system-node-critical" no daemonset do Trident seja atendida. Antes da implantação e da criação do daemonset, Trident procura o `ResourceQuota` objeto e, caso não o encontre, o aplica.

Se você precisar de mais controle sobre a Cota de Recursos e a Classe de Prioridade padrão, você pode gerar um `custom.yaml` ou configurar o `ResourceQuota` objeto usando o Helm chart.

O seguinte é um exemplo de um objeto ResourceQuota priorizando o daemonset Trident.

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
    matchExpressions:
      - operator: In
        scopeName: PriorityClass
        values:
          - system-node-critical
```

Para obter mais informações sobre Resource Quotas, consulte "Kubernetes: cotas de recursos".

Limpe `ResourceQuota` se a instalação falhar

No raro caso em que a instalação falha após o objeto `ResourceQuota` ser criado, tente primeiro "desinstalando" e depois reinstale.

Se isso não funcionar, remova manualmente o `ResourceQuota` objeto.

Remover `ResourceQuota`

Se preferir controlar sua própria alocação de recursos, você pode remover o objeto Trident `ResourceQuota` usando o comando:

```
kubectl delete quota trident-csi -n trident
```

Padrões de Segurança de Pod (PSS) e Restrições de Contexto de Segurança (SCC)

Os padrões de segurança de pods (PSS) e as políticas de segurança de pods (PSP) do Kubernetes definem os níveis de permissão e restringem o comportamento dos pods. OpenShift Security Context Constraints (SCC) definem, de forma semelhante, restrições de pods específicas para o OpenShift Kubernetes Engine. Para fornecer essa personalização, Trident habilita determinadas permissões durante a instalação. As seções a seguir detalham as permissões definidas pelo Trident.

> O PSS substitui as Políticas de Segurança de Pod (PSP). PSP foi descontinuado no Kubernetes v1.21 e será removido na v1.25. Para mais informações, consulte "Kubernetes: segurança".

Contexto de segurança obrigatório do Kubernetes e campos relacionados

Permissão	Descrição
Privilegiado	O CSI exige que os pontos de montagem sejam bidirecionais, o que significa que o pod do nó Trident deve executar um contêiner privilegiado. Para obter mais informações, consulte "Kubernetes: propagação de montagens".
Rede do host	Necessário para o iSCSI daemon. `iscsiadm`Gerencia as montagens iSCSI e usa a rede do host para se comunicar com o iSCSI daemon.
IPC do host	NFS usa comunicação entre processos (IPC) para se comunicar com o NFSD.
PID do host	Necessário para iniciar `rpc-statd` para NFS. Trident consulta os processos do host para determinar se `rpc-statd` está em execução antes de montar volumes NFS.
Capacidades	A `SYS_ADMIN` funcionalidade é fornecida como parte das funcionalidades padrão para contêineres privilegiados. Por exemplo, Docker define essas funcionalidades para contêineres privilegiados: `CapPrm: 0000003fffffffff` `CapEff: 0000003fffffffff`
Seccomp	O perfil Seccomp é sempre "Unconfined" em contêineres privilegiados; portanto, não pode ser ativado no Trident.
SELinux	Em OpenShift, os contêineres privilegiados são executados no `spc_t` ("Super Privileged Container") domínio, e os contêineres não privilegiados são executados no `container_t` domínio. Em `containerd`, com `container-selinux` instalado, todos os contêineres são executados no `spc_t` domínio, o que efetivamente desativa o SELinux. Portanto, Trident não adiciona `seLinuxOptions` aos contêineres.
DAC	Contêineres com privilégios elevados devem ser executados como root. Contêineres sem privilégios elevados são executados como root para acessar os sockets unix necessários pelo CSI.

Padrões de segurança de pod (PSS)

Rótulo	Descrição	Padrão
`pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version`	Permite que o Trident Controller e os nós sejam admitidos no namespace de instalação. Não altere o rótulo do namespace.	`enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>`

Alterar os rótulos de namespace pode resultar em pods não sendo agendados, em um "Error creating: …" ou "Warning: trident-csi-…". Se isso acontecer, verifique se o rótulo do namespace para `privileged` foi alterado. Se sim, reinstale Trident.

Políticas de segurança de pods (PSP)

Campo	Descrição	Padrão
`allowPrivilegeEscalation`	Contêineres privilegiados devem permitir a elevação de privilégios.	`true`
`allowedCSIDrivers`	Trident não utiliza volumes efêmeros CSI inline.	Vazio
`allowedCapabilities`	Os contêineres Trident não privilegiados não exigem mais recursos do que o conjunto padrão e os contêineres privilegiados recebem todas as capacidades possíveis.	Vazio
`allowedFlexVolumes`	Trident não utiliza um "Driver FlexVolume", portanto eles não estão incluídos na lista de volumes permitidos.	Vazio
`allowedHostPaths`	O pod do nó Trident monta o sistema de arquivos raiz do nó, portanto, não há benefício em configurar esta lista.	Vazio
`allowedProcMountTypes`	Trident não usa nenhum `ProcMountTypes`.	Vazio
`allowedUnsafeSysctls`	Trident não requer nenhum `sysctls` inseguro.	Vazio
`defaultAddCapabilities`	Não é necessário adicionar nenhuma funcionalidade aos contêineres privilegiados.	Vazio
`defaultAllowPrivilegeEscalation`	A permissão para escalonamento de privilégios é gerenciada em cada pod do Trident.	`false`
`forbiddenSysctls`	Não `sysctls` são permitidos.	Vazio
`fsGroup`	Os contêineres Trident são executados como root.	`RunAsAny`

Campo	Descrição	Padrão
`hostIPC`	Montar volumes NFS requer IPC do host para se comunicar com `nfsd`	`true`
`hostNetwork`	O iscsiadm requer que a rede do host se comunique com o daemon iSCSI.	`true`
`hostPID`	É necessário o PID do host para verificar se `rpc-statd` está em execução no nó.	`true`
`hostPorts`	Trident não utiliza nenhuma porta do host.	Vazio
`privileged`	Os pods do nó Trident devem executar um contêiner privilegiado para montar volumes.	`true`
`readOnlyRootFilesystem`	Os pods do nó Trident devem gravar no sistema de arquivos do nó.	`false`
`requiredDropCapabilities`	Os pods do nó Trident executam um contêiner privilegiado e não podem descartar recursos.	`none`
`runAsGroup`	Os contêineres Trident são executados como root.	`RunAsAny`
`runAsUser`	Os contêineres Trident são executados como root.	`runAsAny`
`runtimeClass`	Trident não usa `RuntimeClasses`.	Vazio
`seLinux`	Trident não configura `seLinuxOptions` porque atualmente existem diferenças na forma como os ambientes de execução de contêineres e as distribuições do Kubernetes lidam com SELinux.	Vazio
`supplementalGroups`	Os contêineres Trident são executados como root.	`RunAsAny`
`volumes`	Os pods Trident requerem esses plugins de volume.	`hostPath, projected, emptyDir`

Restrições de Contexto de Segurança (SCC)

Etiquetas	Descrição	Padrão
`allowHostDirVolumePlugin`	Os pods do nó Trident montam o sistema de arquivos raiz do nó.	`true`

Etiquetas	Descrição	Padrão
`allowHostIPC`	A montagem de volumes NFS requer IPC do host para se comunicar com `nfsd`.	`true`
`allowHostNetwork`	O iscsiadm requer que a rede do host se comunique com o daemon iSCSI.	`true`
`allowHostPID`	É necessário o PID do host para verificar se `rpc-statd` está em execução no nó.	`true`
`allowHostPorts`	Trident não utiliza nenhuma porta do host.	`false`
`allowPrivilegeEscalation`	Contêineres privilegiados devem permitir a elevação de privilégios.	`true`
`allowPrivilegedContainer`	Os pods do nó Trident devem executar um contêiner privilegiado para montar volumes.	`true`
`allowedUnsafeSysctls`	Trident não requer nenhum `sysctls` inseguro.	`none`
`allowedCapabilities`	Os contêineres Trident não privilegiados não exigem mais recursos do que o conjunto padrão e os contêineres privilegiados recebem todas as capacidades possíveis.	Vazio
`defaultAddCapabilities`	Não é necessário adicionar nenhuma funcionalidade aos contêineres privilegiados.	Vazio
`fsGroup`	Os contêineres Trident são executados como root.	`RunAsAny`
`groups`	Este SCC é específico para Trident e está vinculado ao seu usuário.	Vazio
`readOnlyRootFilesystem`	Os pods do nó Trident devem gravar no sistema de arquivos do nó.	`false`
`requiredDropCapabilities`	Os pods do nó Trident executam um contêiner privilegiado e não podem descartar recursos.	`none`
`runAsUser`	Os contêineres Trident são executados como root.	`RunAsAny`

Etiquetas	Descrição	Padrão
`seLinuxContext`	Trident não configura `seLinuxOptions` porque atualmente existem diferenças na forma como os ambientes de execução de contêineres e as distribuições do Kubernetes lidam com SELinux.	Vazio
`seccompProfiles`	Contêineres privilegiados sempre são executados "Unconfined".	Vazio
`supplementalGroups`	Os contêineres Trident são executados como root.	`RunAsAny`
`users`	É fornecida uma entrada para vincular este SCC ao usuário Trident no namespace Trident.	n/a
`volumes`	Os pods Trident requerem esses plugins de volume.	`hostPath, downwardAPI, projected, emptyDir`

Informações sobre direitos autorais

Informações sobre marcas comerciais