Instale Trident

Trident

NetApp
July 01, 2026

Índice

Instale Trident

Saiba mais sobre a instalação do Trident

Para garantir que Trident possa ser instalado em uma ampla variedade de ambientes e organizações, NetApp oferece múltiplas opções de instalação. Você pode instalar Trident usando o operador Trident (manualmente ou usando Helm) ou com `tridentctl`. Este tópico fornece informações importantes para selecionar o processo de instalação mais adequado para você.

Informações críticas sobre Trident 26.02

Você deve ler as seguintes informações críticas sobre Trident.

Informações críticas sobre Trident

- Kubernetes 1.35 agora é compatível com Trident. Atualize Trident antes de atualizar Kubernetes.
- Trident impõe rigorosamente o uso da configuração de multipath em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração sem multipath ou o uso de `find_multipaths: yes` ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. Trident recomenda o uso de `find_multipaths: no` desde o lançamento 21.07.

Antes de começar

Independentemente do seu caminho de instalação, você deve ter:

- Privilégios completos em um cluster Kubernetes compatível, executando uma versão compatível do Kubernetes e com os requisitos de recursos ativados. Revise a "requisitos" para obter detalhes.
- Acesso a um sistema de storage da NetApp.
- Capacidade de montar volumes a partir de todos os nós de trabalho do Kubernetes.
- Um host Linux com `kubectl` (ou `oc`, se você estiver usando OpenShift) instalado e configurado para gerenciar o cluster Kubernetes que você deseja usar.
- A variável de ambiente `KUBECONFIG` configurada para apontar para a configuração do seu cluster Kubernetes.
- Se você estiver usando Kubernetes com Docker Enterprise, "siga os passos indicados para habilitar o acesso à CLI".
- O cluster deve suportar cargas de trabalho privilegiadas.

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

Escolha o seu método de instalação

Selecione o método de instalação mais adequado para você. Você também deve analisar as considerações para "alternando entre métodos" antes de tomar sua decisão.

Usando o operador Trident

Seja implantando manualmente ou usando o Helm, o operador Trident é uma ótima maneira de simplificar a instalação e gerenciar dinamicamente os recursos do Trident. Você pode até mesmo "Personalize a implantação do seu operador Trident" usando os atributos no `TridentOrchestrator` recurso personalizado (CR).

Os benefícios de usar o operador Trident incluem:

Criação de objeto do Trident

O operador Trident cria automaticamente os seguintes objetos para a sua versão do Kubernetes.

- ServiceAccount para o operador
- ClusterRole e ClusterRoleBinding ao ServiceAccount
- Dedicado PodSecurityPolicy (para Kubernetes 1.25 e versões anteriores)
- O próprio operador

Responsabilidade pelos recursos

O operador Trident com escopo de cluster gerencia os recursos associados a uma instalação do Trident no nível do cluster. Isso mitiga erros que podem ser causados ao manter recursos com escopo de cluster usando um operador com escopo de namespace. Isso é essencial para autorrecuperação e aplicação de patches.

Capacidade de autorrecuperação

O operador monitora a instalação do Trident e toma medidas proativas para resolver problemas, como quando a implantação é excluída ou modificada acidentalmente. Um `trident-operator-<generated-id>` pod é criado que associa um `TridentOrchestrator` CR a uma instalação do Trident. Isso garante que haja apenas uma instância do Trident no cluster e controla sua configuração, assegurando que a instalação seja idempotente. Quando alterações são feitas na instalação (como a exclusão da implantação ou do daemonset do nó), o operador as identifica e as corrige individualmente.

Atualizações fáceis para instalações existentes

Você pode atualizar facilmente 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 Trident para gerar logs de depuração. Para fazer isso, aplique o patch no seu `TridentOrchestrator` para definir `spec.debug` como `true`:

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

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

Reinstalação limpa

O operador Trident com escopo de cluster permite a remoção limpa de recursos com escopo de cluster. Os usuários podem desinstalar completamente o Trident e reinstalá-lo facilmente.

Manipulação automática de upgrade do Kubernetes

Quando a versão do Kubernetes do cluster é atualizada para uma versão compatível, o operador atualiza automaticamente uma instalação existente do Trident e a modifica para garantir que atenda aos requisitos da versão do Kubernetes.

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

Usando `tridentctl`

Se você já possui uma implantação que precisa ser atualizada ou se deseja personalizar sua implantação ao máximo, você deve considerar . Este é o método convencional de implantação do Trident.

Você pode gerar os manifestos para os recursos do Trident. Isso inclui a implantação, o daemonset, a conta de serviço e a função de cluster que Trident cria como parte de sua instalação.

> A partir da versão 22.04, as chaves AES não serão mais regeneradas toda vez que o Trident for instalado. Com esta versão, o Trident instalará um novo objeto secreto que persiste entre as instalações. Isso significa que, `tridentctl` na versão 22.04, é possível desinstalar versões anteriores do Trident, mas versões anteriores não podem desinstalar instalações da 22.04. Selecione o método de instalação apropriado.

Escolha o seu modo de instalação

Determine o processo de implantação com base no (Standard, Offline ou Remote) exigido pela sua organização.

Instalação padrão

Esta é a maneira mais fácil de instalar Trident e funciona na maioria dos ambientes que não impõem restrições de rede. O modo de instalação padrão usa registros padrão para armazenar as imagens necessárias do Trident (`docker.io` e do CSI (`registry.k8s.io`.

Ao usar o modo padrão, o instalador Trident:

- Obtém as imagens do contêiner pela Internet
- Cria um deployment ou daemonset de nó, que inicia pods Trident em todos os nós elegíveis do cluster Kubernetes

Instalação offline

O modo de instalação offline pode ser necessário em um local isolado da internet ou seguro. Nesse cenário, você pode criar um único registro privado espelhado ou dois registros espelhados para armazenar as imagens Trident e CSI necessárias.

Independentemente da configuração do seu registro, as imagens CSI devem residir em um registro.

Instalação remota

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

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

Selecione o processo com base no seu método e modo

Depois de tomar suas decisões, selecione o processo apropriado.

Método	Modo de instalação
Trident operator (manualmente)	"Instalação padrão" "Instalação offline"
Operador Trident (Helm)	"Instalação padrão" "Instalação offline"

Método	Modo de instalação
`tridentctl`	"Instalação padrão ou offline"

Alternando entre métodos de instalação

Você pode optar por alterar seu método de instalação. Antes de fazer isso, considere o seguinte:

- Use sempre o mesmo método para instalar e desinstalar Trident. Se você implantou com `tridentctl`, utilize a versão apropriada do binário `tridentctl` para desinstalar Trident. Da mesma forma, se você estiver implantando com o operador, edite o CR `TridentOrchestrator` e defina `spec.uninstall=true` para desinstalar Trident.
- Se você tiver uma implantação baseada em operador que deseja remover e usar em vez disso `tridentctl` para implantar Trident, primeiro edite `TridentOrchestrator` e defina `spec.uninstall=true` para desinstalar Trident. Em seguida, exclua `TridentOrchestrator` e a implantação do operador. Você pode então instalar usando `tridentctl`.
- Se você possui uma implantação manual baseada em operador e deseja usar a implantação do operador Trident baseada em Helm, você deve desinstalar manualmente o operador primeiro e, em seguida, executar a instalação do Helm. Isso permite que o Helm implante o operador Trident com os rótulos e anotações necessários. Se você não fizer isso, sua implantação do operador Trident baseada em Helm falhará com erro de validação de rótulo e erro de validação de anotação.
- Se você tiver uma implantação baseada em `tridentctl`, poderá realizar implantação baseada em Helm ou em Operador sem desinstalar Trident.

Outras opções de configuração conhecidas

Ao instalar Trident em produtos do VMWare Tanzu Portfolio:

- O `--kubelet-dir` parâmetro deve ser definido para o local do diretório do kubelet. Por padrão, isso é `/var/vcap/data/kubelet`.

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

Instale usando o operador Trident

Implante manualmente o operador Trident (modo padrão)

Você pode implantar manualmente o operador Trident para instalar o Trident. Esse processo se aplica a instalações em que as imagens de contêiner necessárias para o Trident não estão armazenadas em um registro privado. Se você tiver um registro de imagens privado, use o "processo para implantação offline".

Informações críticas sobre Trident 26.02

Você deve ler as seguintes informações críticas sobre Trident.

Informações críticas sobre Trident

- Kubernetes 1.35 agora é compatível com Trident. Atualize Trident antes de atualizar Kubernetes.
- Trident impõe rigorosamente o uso da configuração de multipath em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração sem multipath ou o uso de `find_multipaths: yes` ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. Trident recomenda o uso de `find_multipaths: no` desde o lançamento 21.07.

Implante manualmente o operador Trident e instale o Trident

Revise "visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Antes de iniciar a instalação, faça login no host Linux e verifique se ele está gerenciando um sistema operacional em funcionamento "cluster Kubernetes suportado" e se você possui os privilégios necessários.

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

1. Verifique sua versão do Kubernetes:

```
kubectl version
```

2. Verifique privilégios de administrador do cluster:

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

3. Verifique se você consegue iniciar um pod que utiliza uma imagem do Docker Hub e acessar seu sistema de storage pela rede de pods:

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

Passo 1: Baixe o pacote de instalação do Trident

O pacote de instalação do Trident contém tudo o que você precisa para implantar o operador Trident e instalar o Trident. Baixe e extraia a versão mais recente do instalador do Trident de "a seção *Ativos* em GitHub".

```
wget https://github.com/NetApp/trident/releases/download/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Etapa 2: Criar o `TridentOrchestrator` CRD

Crie a `TridentOrchestrator` Definição de Recurso Personalizado (CRD). Você criará um `TridentOrchestrator` Recurso Personalizado posteriormente. Use a versão YAML apropriada da CRD em `deploy/crds` para criar a `TridentOrchestrator` CRD.

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

Etapa 3: implante o operador Trident

O instalador do Trident fornece um arquivo de pacote que pode ser usado para instalar o operador e criar objetos associados. O arquivo de pacote é uma maneira fácil de implantar o operador e instalar Trident usando uma configuração padrão.

- Para clusters executando Kubernetes 1.24, use `bundle_pre_1_25.yaml`.
- Para clusters executando Kubernetes 1.25 ou posterior, use `bundle_post_1_25.yaml`.

Antes de começar

- Por padrão, o instalador do Trident implanta o operador no `trident` namespace. Se o `trident` namespace não existir, crie-o usando:

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

- Para implantar o operador em um namespace diferente do `trident` namespace, atualize `serviceaccount.yaml`, `clusterrolebinding.yaml` e `operator.yaml` e gere seu arquivo de pacote usando o `kustomization.yaml`.

  a. Crie o `kustomization.yaml` usando o seguinte comando onde *<bundle.yaml>* é `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` com base na sua versão do Kubernetes.

  ```
  cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
  ```

  b. Compile o pacote usando o seguinte comando, onde *<bundle.yaml>* é `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` com base na sua versão do Kubernetes.

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

Passos

1. Crie os recursos e implemente o operador:

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

2. Verifique se o operador, a implantação e os replicasets foram criados.

```
kubectl get all -n <operator-namespace>
```

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

Passo 4: Crie o `TridentOrchestrator` e instale Trident

Agora você pode criar o `TridentOrchestrator` e instalar Trident. Opcionalmente, você pode "Personalize a sua instalação do Trident" usando os atributos na `TridentOrchestrator` spec.

```
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
  nodePrep:
  - iscsi
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:26.02
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:26.02.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v26.02.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Verifique a instalação

Existem várias maneiras de verificar sua instalação.

Usando `TridentOrchestrator` status

O status de `TridentOrchestrator` indica se a instalação foi bem-sucedida e exibe a versão do Trident instalada. Durante a instalação, o status de `TridentOrchestrator` muda de `Installing` para `Installed`. Se você observar o status de `Failed` e o operador não conseguir se recuperar sozinho, "verifique os logs".

Status	Descrição
Instalação	O operador está instalando Trident usando este `TridentOrchestrator` CR.
Instalado	Trident foi instalado com sucesso.
Desinstalando	O operador está desinstalando Trident porque `spec.uninstall=true`.
Desinstalado	Trident foi desinstalado.
Falha	O operador não conseguiu instalar, corrigir, atualizar ou desinstalar Trident; o operador tentará recuperar-se automaticamente deste estado. Se este estado persistir, será necessário realizar a resolução de problemas.
Atualizando	O operador está atualizando uma instalação existente.
Erro	O `TridentOrchestrator` não é usado. Já existe outro.

Usando o status de criação do pod

Você pode confirmar se a instalação do Trident foi concluída verificando o status dos pods criados:

```
kubectl get pods -n trident

NAME                                  READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw   6/6     Running   0
1m
trident-node-linux-mr6zc              2/2     Running   0
1m
trident-node-linux-xrp7w              2/2     Running   0
1m
trident-node-linux-zh2jt              2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv     1/1     Running   0
3m
```

Usando `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

Implantar manualmente o operador Trident (modo offline)

Você pode implantar manualmente o operador Trident para instalar o Trident. Esse processo se aplica a instalações em que as imagens de contêiner necessárias para Trident estão armazenadas em um registro privado. Se você não tiver um registro de imagens privado, use o "processo para implantação padrão".

Informações essenciais sobre Trident

Você deve ler as seguintes informações críticas sobre Trident.

Informações críticas sobre Trident

- Kubernetes 1.35 agora é compatível com Trident. Atualize Trident antes de atualizar Kubernetes.
- Trident impõe rigorosamente o uso da configuração de multipath em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração sem multipath ou o uso de `find_multipaths: yes` ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. Trident recomenda o uso de `find_multipaths: no` desde o lançamento 21.07.

Implante manualmente o operador Trident e instale o Trident

Revise "visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Faça login no host Linux e verifique se ele está gerenciando um sistema operacional em funcionamento "cluster Kubernetes suportado" e se você possui os privilégios necessários.

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

1. Verifique sua versão do Kubernetes:

```
kubectl version
```

2. Verifique privilégios de administrador do cluster:

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

3. Verifique se você consegue iniciar um pod que utiliza uma imagem do Docker Hub e acessar seu sistema de storage pela rede de pods:

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

Passo 1: Baixe o pacote de instalação do Trident

O pacote de instalação do Trident contém tudo o que você precisa para implantar o operador Trident e instalar o Trident. Baixe e extraia a versão mais recente do instalador do Trident de "a seção *Ativos* em GitHub".

```
wget https://github.com/NetApp/trident/releases/download/v6.0/trident-installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Etapa 2: Criar o `TridentOrchestrator` CRD

Crie a `TridentOrchestrator` Definição de Recurso Personalizado (CRD). Você criará um `TridentOrchestrator` Recurso Personalizado posteriormente. Use a versão YAML apropriada da CRD em `deploy/crds` para criar a `TridentOrchestrator` CRD:

```
kubectl create -f deploy/crds/<VERSION>.yaml
```

Etapa 3: atualize o local do registro no operador

Em `/deploy/operator.yaml`, atualize `image: docker.io/netapp/trident-operator:26.02.0` para refletir a localização do seu registro de imagens. Seu "Imagens do Trident e do CSI" pode estar localizado em um registro ou em registros diferentes, mas todas as imagens CSI devem estar no mesmo

registro. Por exemplo:

- `image: <your-registry>/trident-operator:26.02.0` se todas as suas imagens estiverem localizadas em um único registro.
- `image: <your-registry>/netapp/trident-operator:26.02.0` se a sua imagem Trident estiver localizada em um registro diferente das suas imagens CSI.

Etapa 4: implante o operador Trident

O instalador do Trident fornece um arquivo de pacote que pode ser usado para instalar o operador e criar objetos associados. O arquivo de pacote é uma maneira fácil de implantar o operador e instalar Trident usando uma configuração padrão.

- Para clusters executando Kubernetes 1.24, use `bundle_pre_1_25.yaml`.
- Para clusters executando Kubernetes 1.25 ou posterior, use `bundle_post_1_25.yaml`.

Antes de começar

- Por padrão, o instalador do Trident implanta o operador no `trident` namespace. Se o `trident` namespace não existir, crie-o usando:

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

- Para implantar o operador em um namespace diferente do `trident` namespace, atualize `serviceaccount.yaml`, `clusterrolebinding.yaml` e `operator.yaml` e gere seu arquivo de pacote usando o `kustomization.yaml`.

  a. Crie o `kustomization.yaml` usando o seguinte comando onde *<bundle.yaml>* é `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` com base na sua versão do Kubernetes.

  ```
  cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
  ```

  b. Compile o pacote usando o seguinte comando, onde *<bundle.yaml>* é `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` com base na sua versão do Kubernetes.

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

Passos

1. Crie os recursos e implemente o operador:

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

2. Verifique se o operador, a implantação e os replicasets foram criados.

```
kubectl get all -n <operator-namespace>
```

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

Etapa 5: atualize o local do registro de imagens no `TridentOrchestrator`

Seu "Imagens do Trident e do CSI" pode estar localizado em um registro ou em registros diferentes, mas todas as imagens CSI devem estar no mesmo registro. Atualize `deploy/crds/tridentorchestrator_cr.yaml` para adicionar as especificações de localização adicionais com base na sua configuração de registro.

Imagens em um registro

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

Imagens em diferentes repositórios

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

Passo 6: Crie o `TridentOrchestrator` e instale Trident

Agora você pode criar o `TridentOrchestrator` e instalar Trident. Opcionalmente, você pode "Personalize a sua instalação do Trident"personalizar usando os atributos na especificação `TridentOrchestrator`. O exemplo a seguir mostra uma instalação onde as imagens do Trident e do CSI estão localizadas em registros diferentes.

```
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:
  Autosupport Image:  <your-registry>/trident-autosupport:26.02
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.02.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/trident-autosupport:26.02
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/trident:26.02.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v26.02.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Verifique a instalação

Existem várias maneiras de verificar sua instalação.

Usando `TridentOrchestrator` status

O status de `TridentOrchestrator` indica se a instalação foi bem-sucedida e exibe a versão do Trident instalada. Durante a instalação, o status de `TridentOrchestrator` muda de `Installing` para `Installed`. Se você observar o status de `Failed` e o operador não conseguir se recuperar sozinho, "verifique os logs".

Status	Descrição
Instalação	O operador está instalando Trident usando este `TridentOrchestrator` CR.
Instalado	Trident foi instalado com sucesso.
Desinstalando	O operador está desinstalando Trident porque `spec.uninstall=true`.
Desinstalado	Trident foi desinstalado.
Falha	O operador não conseguiu instalar, corrigir, atualizar ou desinstalar Trident; o operador tentará recuperar-se automaticamente deste estado. Se este estado persistir, será necessário realizar a resolução de problemas.
Atualizando	O operador está atualizando uma instalação existente.
Erro	O `TridentOrchestrator` não é usado. Já existe outro.

Usando o status de criação do pod

Você pode confirmar se a instalação do Trident foi concluída verificando o status dos pods criados:

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw        6/6     Running   0
1m
trident-node-linux-mr6zc                   2/2     Running   0
1m
trident-node-linux-xrp7w                   2/2     Running   0
1m
trident-node-linux-zh2jt                   2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv          1/1     Running   0
3m
```

Usando `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

Implantar o operador Trident usando Helm (modo padrão)

Você pode implantar o operador Trident e instalar o Trident usando o Helm. Esse processo se aplica a instalações em que as imagens de contêiner necessárias para o Trident não estão armazenadas em um registro privado. Se você tiver um registro de imagens privado, use o "processo para implantação offline".

Informações críticas sobre Trident 25.10

Você deve ler as seguintes informações críticas sobre Trident.

Informações críticas sobre Trident

- Kubernetes 1.35 agora é compatível com Trident. Atualize Trident antes de atualizar Kubernetes.
- Trident impõe rigorosamente o uso da configuração de multipath em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração sem multipath ou o uso de `find_multipaths: yes` ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. Trident recomenda o uso de `find_multipaths: no` desde o lançamento 21.07.

Implante o operador Trident e instale o Trident usando o Helm

Utilizando o Trident "Helm Chart" você pode implantar o operador Trident e instalar Trident em uma única etapa.

Revise "visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Além do "pré-requisitos de deployment" você precisa de "Helm versão 3".

Passos

1. Adicione o repositório Trident:

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

2. Utilize `helm install` e especifique um nome para sua implantação, como no exemplo a seguir, onde `100..0` é a versão do Trident que você está instalando.

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace <trident-namespace>
```

> Se você já criou um namespace para Trident, o `--create-namespace` parâmetro não criará um namespace adicional.

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

Passe os dados de configuração durante a instalação

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

Opção	Descrição
`--values` (ou `-f`)	Especifique um arquivo YAML com sobrescritas. Isso pode ser especificado várias vezes e o arquivo mais à direita terá precedência.
`--set`	Especifique as substituições na linha de comando.

Por exemplo, para alterar o valor padrão de `debug`, execute o seguinte comando onde `100.2602.0` é a versão do Trident que você está instalando:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set tridentDebug=true
```

Opções de configuração

Esta tabela e o arquivo `values.yaml`, que faz parte do Helm chart, fornecem a lista de chaves e seus valores padrão.

Opção	Descrição	Padrão
`nodeSelector`	Rótulos de nós para atribuição de pods	
`podAnnotations`	Anotações de pod	

Opção	Descrição	Padrão
`deploymentAnnotations`	Anotações de deployment	
`tolerations`	Tolerâncias para atribuição de pod	

`affinity`	Afinidade para atribuição de pod	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedulingIgnoredDuringExecution:`
		`  nodeSelectorTerms:`
		`  - matchExpressions:`
		`  - key: kubernetes.io/arch`
		`  operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key: kubernetes.io/os`
		`  operator: In`
		`  values:`
		`  - linux`
		
		Não remova a afinidade padrão do arquivo values.yaml. Quando desejar fornecer uma afinidade personalizada, estenda a afinidade padrão.

`tridentControllerPluginNodeSelector`	Seletores de nós adicionais para pods. Consulte Entendendo os pods do controlador e os pods do nó para obter detalhes.	
`tridentControllerPluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Consulte Entendendo os pods do controlador e os pods do nó para obter detalhes.	
`tridentNodePluginNodeSelector`	Seletores de nós adicionais para pods. Consulte Entendendo os pods do controlador e os pods do nó para obter detalhes.	

Opção	Descrição	Padrão
`tridentNodePluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Consulte Entendendo os pods do controlador e os pods do nó para obter detalhes.	
`imageRegistry`	Identifica o registro para as `trident-operator`, `trident` e outras imagens. Deixe em branco para aceitar o padrão. IMPORTANTE: Ao instalar Trident em um repositório privado, se você estiver usando o `imageRegistry` para especificar a localização do repositório, não use `/netapp/` no caminho do repositório.	`""`
`imagePullPolicy`	Define a política de pull de imagens para o `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Define os segredos de extração de imagens para as `trident-operator`, `trident` e outras imagens.	
`kubeletDir`	Permite substituir a localização do host do estado interno do kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Permite definir o nível de log do operador Trident para: `trace`, `debug`, `info`, `warn`, `error`, ou `fatal`.	`"info"`
`operatorDebug`	Permite definir o nível de registro do operador Trident como depuração.	`true`
`operatorImage`	Permite a substituição completa da imagem para `trident-operator`.	`""`
`operatorImageTag`	Permite sobrescrever a tag da `trident-operator` imagem.	`""`
`tridentIPv6`	Permite habilitar Trident para funcionar em clusters IPv6.	`false`
`tridentK8sTimeout`	Substitui o tempo limite padrão de 30 segundos para a maioria das operações da API do Kubernetes (se diferente de zero, em segundos).	`0`

Opção	Descrição	Padrão
`tridentHttpRequestTimeout`	Substitui o tempo limite padrão de 90 segundos para as solicitações HTTP, com `0s` sendo uma duração infinita para o tempo limite. Valores negativos não são permitidos.	`"90s"`
`tridentSilenceAutosupport`	Permite desativar o relatório periódico AutoSupport do Trident.	`false`
`tridentAutosupportImageTag`	Permite substituir a tag da imagem para o contêiner Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Permite que o contêiner Trident AutoSupport se comunique com um servidor externo via um proxy HTTP.	`""`
`tridentLogFormat`	Define o formato de registro do Trident (`text` ou `json`).	`"text"`
`tridentDisableAuditLog`	Desativa o registrador de auditoria do Trident.	`true`
`tridentLogLevel`	Permite definir o nível de log do Trident para: `trace`, `debug`, `info`, `warn`, `error`, ou `fatal`.	`"info"`
`tridentDebug`	Permite definir o nível de log do Trident para `debug`. Você pode automatizar o processo de desanexação forçada por meio da integração com o operador de verificação de integridade do nó (NHC). Para obter informações, consulte "Automatizando o failover de aplicações stateful com Trident".	`false`
`tridentLogWorkflows`	Permite que fluxos de trabalho específicos do Trident sejam habilitados para registro de rastreamento ou supressão de logs.	`""`
`tridentLogLayers`	Permite que camadas específicas do Trident sejam ativadas para registro de rastreamento ou supressão de logs.	`""`
`tridentImage`	Permite a substituição completa da imagem para Trident.	`""`
`tridentImageTag`	Permite substituir a tag da imagem para Trident.	`""`

Opção	Descrição	Padrão
`tridentProbePort`	Permite substituir a porta padrão usada para as sondagens de liveness/readiness do Kubernetes.	`""`
`windows`	Permite que o Trident seja instalado em nó de trabalho Windows.	`false`
`enableForceDetach`	Permite ativar o recurso de force detach.	`false`
`excludePodSecurityPolicy`	Exclui a política de segurança do pod do operador da criação.	`false`
`cloudProvider`	Defina como `"Azure"` ao usar identidades gerenciadas ou uma identidade de nuvem em um cluster AKS. Defina como "AWS" ao usar uma identidade de nuvem em um cluster EKS.	`""`
`cloudIdentity`	Defina como identidade da carga de trabalho ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx") ao usar identidade da nuvem em um cluster AKS. Defina como AWS IAM role ("'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role'") ao usar identidade da nuvem em um cluster EKS.	`""`
`iscsiSelfHealingInterval`	O intervalo em que a autorrecuperação iSCSI é acionada.	`5m0s`
`iscsiSelfHealingWaitTime`	A duração após a qual a autorrecuperação iSCSI inicia uma tentativa de resolver uma sessão inativa realizando um logout e, em seguida, um login.	`7m0s`
`nodePrep`	Permite que Trident prepare os nós do cluster Kubernetes para gerenciar volumes usando o protocolo de storage de dados especificado. **Atualmente, `iscsi` é o único valor suportado.** NOTA: A partir de OpenShift 4.19, a versão mínima do Trident compatível com este recurso é 25.06.1.	

Opção	Descrição	Padrão
`enableConcurrency`	Permite operações simultâneas do controlador Trident para melhorar o desempenho. **Prévia Técnica**: Este recurso é experimental e atualmente suporta fluxos de trabalho paralelos limitados com os drivers ONTAP-NAS (somente NFS) e ONTAP-SAN (NVMe para ONTAP 9 unificado), além da prévia técnica existente para o driver ONTAP-SAN (protocolos iSCSI e FCP no ONTAP 9 unificado).	falso
`k8sAPIQPS`	O limite de consultas por segundo (QPS) usado pelo controlador ao se comunicar com o servidor da API do Kubernetes. O valor de Burst é definido automaticamente com base no valor de QPS.	`100`; opcional

Opção	Descrição	Padrão

`resources`	Define os limites e solicitações de recursos do Kubernetes para os pods do Trident controller, node e operator. Você pode configurar CPU e memória para cada contêiner e sidecar para gerenciar a alocação de recursos no Kubernetes.
	
	Para mais informações sobre como configurar solicitações e limites de recursos, consulte "Gerenciamento de recursos para pods e contêineres".
	
	• NÃO altere os nomes de nenhum container ou campo.
	• NÃO altere a indentação - a indentação do YAML é fundamental para a análise correta.
		`resources:`
		`  controller:`
		`  trident-main:`
		`  requests:`
		`  cpu: 10m`
		`  memory: 80Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-provisioner:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-attacher:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-resizer:`
		`  requests:`
		`  cpu: 3m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-snapshotter:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  trident-autosupport:`
		`  requests:`
		`  cpu: 1m`
		`  memory: 30Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  node:`
		`  linux:`
		`  trident-main:`

Opção	Descrição	Padrão
`httpsMetrics`	Habilite o HTTPS para o endpoint de métricas do Prometheus.	falso
`hostNetwork`	Habilita a rede do host para o controlador Trident. Isso é útil quando você deseja separar o tráfego de frontend e backend em uma rede com múltiplas interfaces.	falso

automaticament
e caso não
sejam
especificadas

- Os nomes dos contêineres são listados conforme aparecem nas especificações do pod.
- Os sidecars estão listados sob cada contêiner principal.
- Verifique o campo `status.lastOperationStatus` em `tridentOrchestrator` para visualizar os valores atualmente aplicados.

```
requests:
  cpu: 1m
  memory: 10Mi
limits:
  cpu:
  memory:
windows:
  trident-main:
    requests:
      cpu: 6m
      memory: 40Mi
    limits:
      cpu:
      memory:
  node-driver-registrar:
    requests:
      cpu: 6m
      memory: 40Mi
    limits:
      cpu:
      memory:
  liveness-probe:
    requests:
      cpu: 2m
      memory: 40Mi
```

Entendendo os pods do controlador e os pods do nó

Trident é executado como um único pod controlador, além de um pod de nó em cada nó de trabalho do cluster. O pod de nó deve estar em execução em qualquer host onde você deseje montar um volume do Trident.

Kubernetes "seletores de nós" e "tolerations e taints" são usados para restringir um pod a ser executado em um nó específico ou preferencial. Usando o `ControllerPlugin` e `NodePlugin`, você pode especificar restrições e substituições.

- O plugin do controlador lida com o provisionamento e gerenciamento de volumes, como snapshots e redimensionamento.
- O plugin do nó gerencia a conexão do storage ao nó.

Implantar o operador Trident usando Helm (modo offline)

Você pode implantar o operador Trident e instalar o Trident usando o Helm. Esse processo se aplica a instalações em que as imagens de contêiner necessárias para Trident estão armazenadas em um registro privado. Se você não tiver um registro de imagens privado, use o "processo para implantação padrão".

Informações críticas sobre Trident 26.02

Você deve ler as seguintes informações críticas sobre Trident.

Informações críticas sobre Trident

- Kubernetes 1.35 agora é compatível com Trident. Atualize Trident antes de atualizar Kubernetes.
- Trident impõe rigorosamente o uso da configuração de multipath em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração sem multipath ou o uso de `find_multipaths: yes` ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. Trident recomenda o uso de `find_multipaths: no` desde o lançamento 21.07.

Implante o operador Trident e instale o Trident usando o Helm

Utilizando o Trident "Helm Chart" você pode implantar o operador Trident e instalar Trident em uma única etapa.

Revise "visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Além do "pré-requisitos de deployment" você precisa de "Helm versão 3".

> Ao instalar Trident em um repositório privado, se você estiver usando o `imageRegistry` switch para especificar a localização do repositório, não use `/netapp/` no caminho do repositório.

Passos

1. Adicione o repositório Trident:

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

2. Use `helm install` e especifique um nome para sua implantação e o local do registro de imagens. Seu "Imagens do Trident e do CSI" pode estar localizado em um registro ou em registros diferentes, mas todas as imagens CSI devem estar no mesmo registro. Nos exemplos, `100.2602.0` é a versão do Trident que você está instalando.

Imagens em um registro

```
helm install <name> netapp-trident/trident-operator --version
100.2602.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace> --set nodePrep={iscsi}
```

Imagens em diferentes repositórios

```
helm install <name> netapp-trident/trident-operator --version
100.2602.0 --set imageRegistry=<your-registry> --set operatorImage
=<your-registry>/trident-operator:26.02.0 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:26.02
--set tridentImage=<your-registry>/trident:26.02.0 --create
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```

> Se você já criou um namespace para Trident, o `--create-namespace` parâmetro não criará um namespace adicional.

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

Passe os dados de configuração durante a instalação

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

Opção	Descrição
`--values` (ou `-f`)	Especifique um arquivo YAML com sobrescritas. Isso pode ser especificado várias vezes e o arquivo mais à direita terá precedência.
`--set`	Especifique as substituições na linha de comando.

Por exemplo, para alterar o valor padrão de `debug`, execute o seguinte comando onde `100.2602.0` é a versão do Trident que você está instalando:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set tridentDebug=true
```

Para adicionar o valor nodePrep, execute o seguinte comando:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set nodePrep={iscsi}
```

Opções de configuração

Esta tabela e o arquivo `values.yaml`, que faz parte do Helm chart, fornecem a lista de chaves e seus valores padrão.

> Não remova a afinidade padrão do arquivo values.yaml. Quando desejar fornecer uma afinidade personalizada, estenda a afinidade padrão.

Opção	Descrição	Padrão
`nodeSelector`	Rótulos de nós para atribuição de pods	
`podAnnotations`	Anotações de pod	
`deploymentAnnotations`	Anotações de deployment	
`tolerations`	Tolerâncias para atribuição de pod	

Opção	Descrição	Padrão

`affinity`	Afinidade para atribuição de pod	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedul`
		`ingIgnoredDuringExecu`
		`tion:`
		
		`nodeSelectorTerms:`
		`  -`
		`matchExpressions:`
		`  - key:`
		`kubernetes.io/arch`
		
		`operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key:`
		`kubernetes.io/os`
		
		`operator: In`
		`  values:`
		`  - linux`
		
		Não remova a afinidade padrão do arquivo values.yaml. Quando desejar fornecer uma afinidade personalizada, estenda a afinidade padrão.

`tridentControllerPluginNodeSelector`	Seletores de nós adicionais para pods. Consulte "Entendendo os pods do controlador e os pods do nó" para obter detalhes.	
`tridentControllerPluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Consulte "Entendendo os pods do controlador e os pods do nó" para obter detalhes.	

Opção	Descrição	Padrão
`tridentNodePluginNodeSelector`	Seletores de nós adicionais para pods. Consulte "Entendendo os pods do controlador e os pods do nó" para obter detalhes.	
`tridentNodePluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Consulte "Entendendo os pods do controlador e os pods do nó" para obter detalhes.	
`imageRegistry`	Identifica o registro para as `trident-operator`, `trident` e outras imagens. Deixe em branco para aceitar o padrão. IMPORTANTE: Ao instalar Trident em um repositório privado, se você estiver usando o `imageRegistry` para especificar a localização do repositório, não use `/netapp/` no caminho do repositório.	""
`imagePullPolicy`	Define a política de pull de imagens para o `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Define os segredos de extração de imagens para as `trident-operator`, `trident` e outras imagens.	
`kubeletDir`	Permite substituir a localização do host do estado interno do kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Permite definir o nível de log do operador Trident para: `trace`, `debug`, `info`, `warn`, `error`, ou `fatal`.	`"info"`
`operatorDebug`	Permite definir o nível de registro do operador Trident como depuração.	`true`
`operatorImage`	Permite a substituição completa da imagem para `trident-operator`.	""
`operatorImageTag`	Permite sobrescrever a tag da `trident-operator` imagem.	""
`tridentIPv6`	Permite habilitar Trident para funcionar em clusters IPv6.	`false`

Opção	Descrição	Padrão
`tridentK8sTimeout`	Substitui o tempo limite padrão de 180 segundos para a maioria das operações da API do Kubernetes (se diferente de zero, em segundos). O `tridentK8sTimeout` parâmetro aplica-se apenas à instalação do Trident.	`180`
`tridentHttpRequestTimeout`	Substitui o tempo limite padrão de 90 segundos para as solicitações HTTP, com `0s` sendo uma duração infinita para o tempo limite. Valores negativos não são permitidos.	`"90s"`
`tridentSilenceAutosupport`	Permite desativar o relatório periódico AutoSupport do Trident.	`false`
`tridentAutosupportImageTag`	Permite substituir a tag da imagem para o contêiner Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Permite que o contêiner Trident AutoSupport se comunique com um servidor externo via um proxy HTTP.	""
`tridentLogFormat`	Define o formato de registro do Trident (`text` ou `json`).	`"text"`
`tridentDisableAuditLog`	Desativa o registrador de auditoria do Trident.	`true`
`tridentLogLevel`	Permite definir o nível de log do Trident para: `trace`, `debug`, `info`, `warn`, `error`, ou `fatal`.	`"info"`
`tridentDebug`	Permite definir o nível de log do Trident para `debug`.	`false`
`tridentLogWorkflows`	Permite que fluxos de trabalho específicos do Trident sejam habilitados para registro de rastreamento ou supressão de logs.	""
`tridentLogLayers`	Permite que camadas específicas do Trident sejam ativadas para registro de rastreamento ou supressão de logs.	""

Opção	Descrição	Padrão
`tridentImage`	Permite a substituição completa da imagem para Trident.	""
`tridentImageTag`	Permite substituir a tag da imagem para Trident.	""
`tridentProbePort`	Permite substituir a porta padrão usada para as sondagens de liveness/readiness do Kubernetes.	""
`windows`	Permite que o Trident seja instalado em nó de trabalho Windows.	`false`
`enableForceDetach`	Permite ativar o recurso de force detach. Você pode automatizar o processo de desanexação forçada por meio da integração com o operador de verificação de integridade do nó (NHC). Para obter informações, consulte "Automatizando o failover de aplicações stateful com Trident".	`false`
`excludePodSecurityPolicy`	Exclui a política de segurança do pod do operador da criação.	`false`
`nodePrep`	Permite que Trident prepare os nós do cluster Kubernetes para gerenciar volumes usando o protocolo de storage de dados especificado. **Atualmente, `iscsi` é o único valor suportado.** A partir do OpenShift 4.19, a versão mínima do Trident compatível com este recurso é 25.06.1.	

Opção	Descrição	Padrão

`resources`	Define os limites e solicitações de recursos do Kubernetes para os pods do Trident controller, node e operator. Você pode configurar CPU e memória para cada contêiner e sidecar para gerenciar a alocação de recursos no Kubernetes.
	
	Para mais informações sobre como configurar solicitações e limites de recursos, consulte "Gerenciamento de recursos para pods e contêineres".
	
	• NÃO altere os nomes de nenhum container ou campo.
	• NÃO altere a indentação - a indentação do YAML é fundamental para a análise correta.
		`resources:`
		`  controller:`
		`  trident-main:`
		`  requests:`
		`  cpu: 10m`
		`  memory: 80Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-provisioner:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-attacher:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-resizer:`
		`  requests:`
		`  cpu: 3m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-snapshotter:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  trident-autosupport:`
		`  requests:`
		`  cpu: 1m`
		`  memory: 30Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  node:`
		`  linux:`

Personalizar instalação do Trident

O operador Trident permite que você personalize a instalação do Trident usando os atributos na `TridentOrchestrator`spec. Se você quiser personalizar a instalação além do que os `TridentOrchestrator`argumentos permitem, considere usar `tridentctl`para gerar manifestos YAML personalizados para modificar conforme necessário.

Entendendo os pods do controlador e os pods do nó

Trident é executado como um único pod de controlador e um pod de nó em cada nó de trabalho do cluster. O pod de nó deve estar em execução em qualquer host onde você deseje montar um volume do Trident.

Kubernetes "seletores de nós" e "tolerations e taints" são usados para restringir um pod a ser executado em um nó específico ou preferencial. Usando o `ControllerPlugin` e `NodePlugin`, você pode especificar restrições e substituições.

- O plugin do controlador lida com o provisionamento e gerenciamento de volumes, como snapshots e redimensionamento.
- O plugin do nó gerencia a conexão do storage ao nó.

Opções de configuração

> **Importante**
>
> `spec.namespace` é especificado em `TridentOrchestrator` para indicar o namespace onde Trident está instalado. Este parâmetro **não pode ser atualizado após Trident ser instalado**. Tentar fazer isso faz com que o status `TridentOrchestrator` mude para `Failed`. Trident não foi projetado para ser migrado entre namespaces.

Esta tabela detalha `TridentOrchestrator` atributos.

Parâmetro	Descrição	Padrão
`namespace`	Namespace para instalar Trident	`"default"`
`debug`	Ative a depuração para Trident	`false`
`enableForceDetach`	`ontap-san`, `ontap-san-economy`, `ontap-nas`, e `ontap-nas-economy` somente. Funciona com Kubernetes Non-Graceful Node Shutdown (NGNS) para conceder aos administradores do cluster a capacidade de migrar cargas de trabalho com volumes montados para novos nós com segurança, caso um nó fique indisponível. Para obter informações, consulte "Automatizando o failover de aplicações stateful com Trident".	`false`
`windows`	Definir como `true` permite instalação em nós de trabalho Windows.	`false`

- Por padrão, não há limites aplicados - apenas as solicitações possuem valores padrão.
- Os nomes dos contêineres são listados conforme aparecem nas especificações do pod.
- Os sidecars estão listados sob cada contêiner principal.
- Verifique o campo `status` dos CRs do TORC para visualizar os valores atualmente

```
      trident-main:
        requests:
          cpu: 10m
          memory:
60Mi
        limits:
          cpu:
          memory:
      node-driver-
registrar:
        requests:
          cpu: 1m
          memory:
10Mi
        limits:
          cpu:
          memory:
    windows:
      trident-main:
        requests:
          cpu: 6m
          memory:
40Mi
        limits:
          cpu:
          memory:
  operator:
    requests:
      cpu: 10m
      memory: 40Mi
    limits:
```

Parâmetro	Descrição	Padrão
`cloudProvider`	Defina como `"Azure"` ao usar identidades gerenciadas ou uma identidade de nuvem em um cluster AKS. Defina como `"AWS"` ao usar uma identidade de nuvem em um cluster EKS. Defina como `"GCP"` ao usar uma identidade de nuvem em um cluster GKE.	`""`
`cloudIdentity`	Defina como identidade da carga de trabalho ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx") ao usar identidade da nuvem em um cluster AKS. Defina como AWS IAM role ("'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role'") ao usar identidade da nuvem em um cluster EKS. Defina como cloud identity ("'iam.gke.io/gcp-service-account: xxxx@mygcpproject.iam.gserviceaccount.com'") ao usar cloud identity em um cluster GKE.	`""`
`IPv6`	Instale o Trident via IPv6	falso
`k8sTimeout`	Tempo limite para operações do Kubernetes. O `k8sTimeout`parâmetro aplica-se apenas à instalação do Trident.	`180sec`
`silenceAutosupport`	Não envie pacotes de autosupport para a NetApp automaticamente	`false`
`autosupportImage`	A imagem do contêiner para Autosupport Telemetry	`"netapp/trident-autosupport10"`
`autosupportProxy`	O endereço/porta de um proxy para envio de telemetria do Autosupport	`"http://proxy.example.com:8888"`
`uninstall`	Uma flag usada para desinstalar Trident	`false`
`logFormat`	Formato de registro do Trident a ser usado [texto,json]	`"text"`
`tridentImage`	Imagem Trident para instalar	`"netapp/trident:26.02"`
`imageRegistry`	Caminho para o registro interno, no formato `<registry FQDN>[:port][/subpath]`	`"registry.k8s.io"`
`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 Trident	
`imagePullSecrets`	Segredos para puxar imagens de um registro interno	

Parâmetro	Descrição	Padrão
`imagePullPolicy`	Define a política de pull de imagem para o operador Trident. Os valores válidos são: `Always` para sempre puxar a imagem. `IfNotPresent` para puxar a imagem somente se ela ainda não existir no nó. `Never` para nunca puxar a imagem.	`IfNotPresent`
`controllerPluginNodeSelector`	Seletores de nós adicionais para pods. Segue o mesmo formato que `pod.spec.nodeSelector`.	Sem valor padrão; opcional
`controllerPluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Segue o mesmo formato que `pod.spec.Tolerations`.	Sem valor padrão; opcional
`nodePluginNodeSelector`	Seletores de nós adicionais para pods. Segue o mesmo formato que `pod.spec.nodeSelector`.	Sem valor padrão; opcional
`nodePluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Segue o mesmo formato que `pod.spec.Tolerations`.	Sem valor padrão; opcional
`nodePrep`	Permite que Trident prepare os nós do cluster Kubernetes para gerenciar volumes usando o protocolo de storage de dados especificado. **Atualmente, `iscsi` é o único valor suportado.** A partir do OpenShift 4.19, a versão mínima do Trident compatível com este recurso é 25.06.1.	
`k8sAPIQPS`	O limite de consultas por segundo (QPS) usado pelo controlador ao se comunicar com o servidor da API do Kubernetes. O valor de Burst é definido automaticamente com base no valor de QPS.	`100`; opcional
`enableConcurrency`	Permite operações simultâneas do controlador Trident para melhorar o desempenho. **Prévia Técnica**: Este recurso é experimental e atualmente suporta fluxos de trabalho paralelos limitados com os drivers ONTAP-NAS (somente NFS) e ONTAP-SAN (NVMe para ONTAP 9 unificado), além da prévia técnica existente para o driver ONTAP-SAN (protocolos iSCSI e FCP no ONTAP 9 unificado).	falso

Parâmetro	Descrição	Padrão

`resources`	Define os limites de recursos e as solicitações do Kubernetes para o controlador Trident e os pods dos nós. Você pode configurar CPU e memória para cada contêiner e sidecar para gerenciar a alocação de recursos no Kubernetes.
	
	Para mais informações sobre como configurar solicitações e limites de recursos, consulte "Gerenciamento de recursos para pods e contêineres".
	
	• NÃO altere os nomes de nenhum container ou campo.
	• NÃO altere a indentação - a indentação do YAML é fundamental para a análise correta.
	
	• Por padrão, não há limites aplicados - apenas as solicitações possuem valores padrão e são aplicadas automaticamente caso não sejam especificadas.
	• Os nomes dos contêineres são listados conforme aparecem nas especificações do pod.
	• Os sidecars estão listados sob cada contêiner principal.
	• Verifique o campo `status.CurrentInstallationParams` do TORC para visualizar os valores atualmente aplicados.
		`resources:`
		`  controller:`
		`    trident-main:`
		`      requests:`
		`        cpu: 10m`
		`        memory: 80Mi`
		`      limits:`
		`          cpu:`
		`memory:`
		`    csi-provisioner:`
		`      requests:`
		`        cpu: 2m`
		`        memory: 20Mi`
		`      limits:`
		`        cpu:`
		`        memory:`
		`    csi-attacher:`
		`      requests:`
		`        cpu: 2m`
		`        memory: 20Mi`
		`      limits:`
		`        cpu:`
		`        memory:`
		`    csi-resizer:`
		`      requests:`
		`        cpu: 3m`
		`        memory: 20Mi`
		`      limits:`
		`        cpu:`
		`        memory:`
		`    csi-snapshotter:`
		`      requests:`
		`        cpu: 2m`
		`        memory: 20Mi`
		`      limits:`

Parâmetro	Descrição	Padrão
`httpsMetrics`	Habilite o HTTPS para o endpoint de métricas do Prometheus.	falso
`hostNetwork`	Habilita a rede do host para o controlador Trident. Isso é útil quando você deseja separar o tráfego de frontend e backend em uma rede com múltiplas interfaces.	falso

Para mais informações sobre formatação de parâmetros de pod, consulte "Atribuindo Pods a Nodes".

Configurações de exemplo

Você pode usar os atributos em Opções de configuração ao definir `TridentOrchestrator` para personalizar sua instalação.

Configuração personalizada básica

Este exemplo, criado após a execução do `cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml` comando, representa uma instalação personalizada básica:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
```

Seletores de nós

Este exemplo instala Trident 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
```

Nós de trabalho do Windows

Este exemplo, criado após a execução do `cat deploy/crds/tridentorchestrator_cr.yaml` comando, instala Trident em um nó de trabalho do Windows.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  windows: true
```

Identidades gerenciadas em um cluster AKS

Este exemplo instala Trident para habilitar identidades gerenciadas em um cluster AKS.

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

Identidade na nuvem em um cluster AKS

Este exemplo instala Trident para uso com uma identidade na nuvem em um cluster AKS.

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

Identidade na nuvem em um cluster EKS

Este exemplo instala Trident para uso com uma identidade na nuvem em um cluster AKS.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/trident-role'"
```

Identidade na nuvem para GKE

Este exemplo instala Trident para uso com uma identidade na nuvem em um cluster GKE.

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

Configuração de solicitações e limites de recursos do Kubernetes para o controlador Trident e pods de nós Linux do Trident

Este exemplo configura as solicitações e limites de recursos do Kubernetes para os pods do controlador Trident e do nó Linux do Trident.

Aviso: Os valores de solicitação e limite fornecidos neste exemplo são apenas para fins de demonstração. Ajuste esses valores de acordo com seu ambiente e requisitos de carga de trabalho.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
      # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
          cpu: 3m
```

```
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
            memory: 60Mi
          limits:
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
            cpu: 50m
            memory: 32Mi
```

Configuração de solicitações e limites de recursos do Kubernetes para o controlador Trident e pods de nós Trident Windows e Linux

Este exemplo configura solicitações e limites de recursos do Kubernetes para o controlador Trident e para os pods dos nós Trident Windows e Linux.

Aviso: Os valores de solicitação e limite fornecidos neste exemplo são apenas para fins de demonstração. Ajuste esses valores de acordo com seu ambiente e requisitos de carga de trabalho.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  windows: true
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
        # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
```

```
          cpu: 3m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
            memory: 60Mi
          limits:
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
            cpu: 50m
            memory: 32Mi
      windows:
        trident-main:
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 200m
            memory: 128Mi
        # sidecars
        node-driver-registrar:
```

```
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 100m
            memory: 128Mi
        liveness-probe:
          requests:
            cpu: 2m
            memory: 40Mi
          limits:
            cpu: 50m
            memory: 64Mi
```

Instalar usando o tridentctl

Instalar usando o tridentctl

Você pode instalar Trident usando `tridentctl`. Este processo se aplica a instalações onde as imagens de contêiner necessárias para Trident estão armazenadas em um registro privado ou não. Para personalizar sua `tridentctl` implantação, consulte "Personalize a implantação do tridentctl".

Informações críticas sobre Trident10

Você deve ler as seguintes informações críticas sobre Trident.

Informações críticas sobre Trident

- Kubernetes 1.27 agora é compatível com Trident. Atualize Trident antes de atualizar Kubernetes.
- Trident impõe rigorosamente o uso da configuração de multipath em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração sem multipath ou o uso de `find_multipaths: yes` ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. Trident recomenda o uso de `find_multipaths: no` desde o lançamento 21.07.

Instale Trident usando `tridentctl`

Revise "visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Antes de iniciar a instalação, faça login no host Linux e verifique se ele está gerenciando um sistema operacional em funcionamento "cluster Kubernetes suportado" e se você possui os privilégios necessários.

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

1. Verifique sua versão do Kubernetes:

```
kubectl version
```

2. Verifique privilégios de administrador do cluster:

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

3. Verifique se você consegue iniciar um pod que utiliza uma imagem do Docker Hub e acessar seu sistema de storage pela rede de pods:

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

Passo 1: Baixe o pacote de instalação do Trident

O pacote de instalação do Trident cria um pod do Trident, configura os objetos CRD que são usados para manter seu estado e inicializa os sidecars CSI para executar ações como provisionar e anexar volumes aos hosts do cluster. Baixe e extraia a versão mais recente do instalador do Trident de "a seção *Ativos* em GitHub". Atualize *<trident-installer-XX.XX.X.tar.gz>* no exemplo com a versão do Trident selecionada.

```
wget https://github.com/NetApp/trident/releases/download/v26.02.0/trident-installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Passo 2: Instalar Trident

Instale Trident no namespace desejado executando o comando `tridentctl install`. Você pode adicionar argumentos adicionais para especificar o local do registro de imagens.

Modo padrão

```
./tridentctl install -n trident
```

Imagens em um registro

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.02 --trident
-image <your-registry>/trident:26.02.0
```

Imagens em diferentes repositórios

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.02 --trident
-image <your-registry>/trident:26.02.0
```

O status da sua instalação deve ser semelhante a este.

```
....
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-controller-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=26.10.0
INFO Trident installation succeeded.
....
```

Verifique a instalação

Você pode verificar sua instalação usando o status de criação do pod ou `tridentctl`.

Usando o status de criação do pod

Você pode confirmar se a instalação do Trident foi concluída verificando o status dos pods criados:

```
kubectl get pods -n trident

NAME                                  READY   STATUS    RESTARTS   AGE
trident-controller-679648bd45-cv2mx   6/6     Running   0          5m29s
trident-node-linux-vgc8n              2/2     Running   0          5m29s
```

Se o instalador não for concluído com êxito ou `trident-controller-<generated id>` (`trident-csi-<generated id>` em versões anteriores à 23.01) não apresentar o status **Running**, a plataforma não foi instalada. Use `-d` para "ative o modo de depuração" e solucionar o problema.

Usando `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

Configurações de exemplo

Os exemplos a seguir fornecem configurações de exemplo para instalar Trident usando `tridentctl`.

Nós Windows

Para habilitar o Trident para ser executado em nós Windows:

```
tridentctl install --windows -n trident
```

Forçar desanexação

Para obter informações, consulte "Automatizando o failover de aplicações stateful com Trident".

```
tridentctl install --enable-force-detach=true -n trident
```

Habilitar operações simultâneas do controlador Trident

Para habilitar operações simultâneas do controlador Trident e melhorar o desempenho, adicione a opção `--enable-concurrency` durante a instalação, conforme mostrado neste exemplo.

> **Prévia Técnica**: Este recurso é experimental e atualmente suporta fluxos de trabalho paralelos limitados com os drivers ONTAP-NAS (somente NFS) e ONTAP-SAN (NVMe para ONTAP 9 unificado), além da prévia técnica existente para o driver ONTAP-SAN (protocolos iSCSI e FCP no ONTAP 9 unificado).

```
tridentctl install --enable-concurrency -n trident
```

Personalize a instalação do tridentctl

Você pode usar o instalador Trident para personalizar a instalação.

Saiba mais sobre o instalador

O instalador do Trident permite que você personalize atributos. Por exemplo, se você copiou a imagem do Trident para um repositório privado, pode especificar o nome da imagem usando `--trident-image`. Se você copiou a imagem do Trident, assim como as imagens auxiliares CSI necessárias, para um repositório privado, pode ser preferível especificar a localização desse repositório usando o comando `--image-registry`, que tem o formato `<registry FQDN>[:port]`.

> Ao instalar Trident em um repositório privado, se você estiver usando o `--image-registry` switch para especificar a localização do repositório, não use `/netapp/` no caminho do repositório. Por exemplo: `./tridentctl install --image-registry <image-registry> -n <namespace>`

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

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

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

- `trident-serviceaccount.yaml`
- `trident-resourcequota.yaml` *

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

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

Instale usando o operador certificado OpenShift

Instale Trident usando OpenShift OperatorHub

Se você usa o Red Hat OpenShift, pode instalar NetApp Trident usando o operador certificado da Red Hat. Use este procedimento para instalar Trident a partir da Red Hat OpenShift Container Platform.

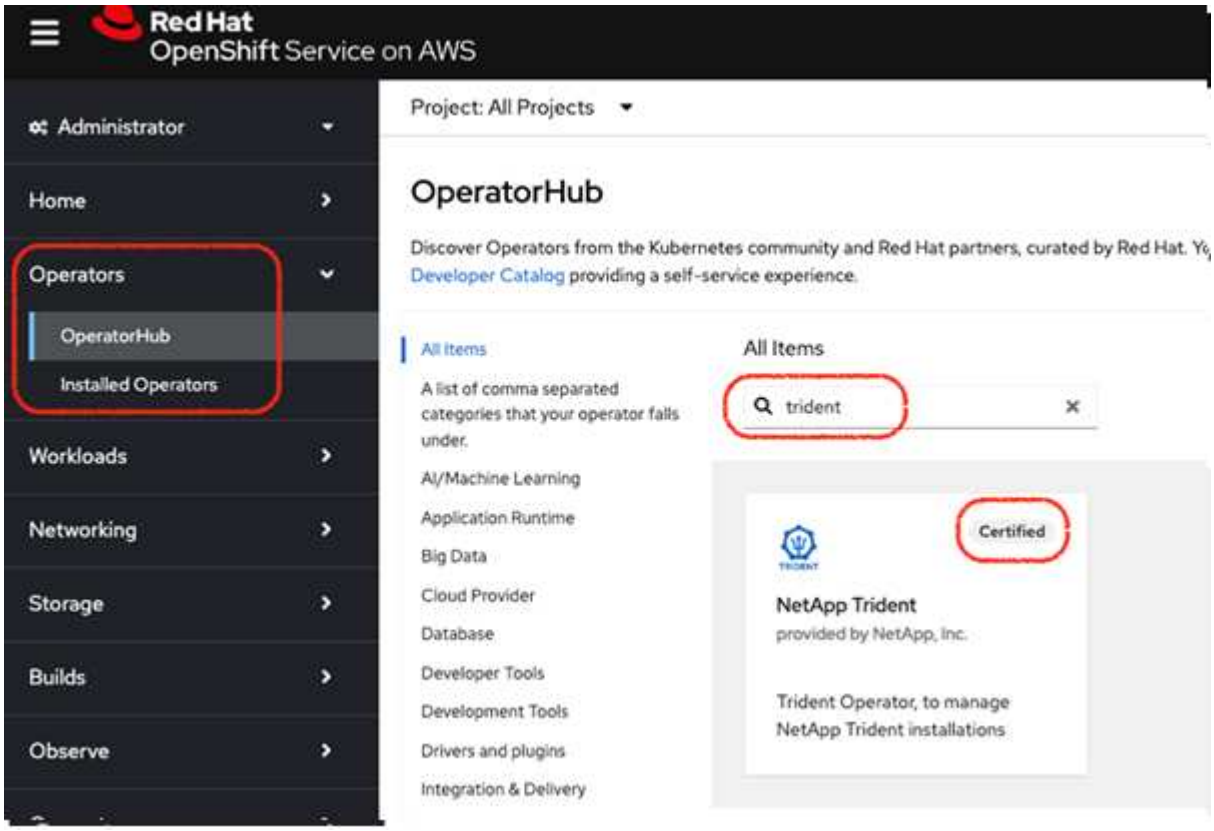
Antes de começar

Antes de iniciar a instalação, "prepare seu ambiente para a instalação do Trident".

Localize e instale o operador Trident

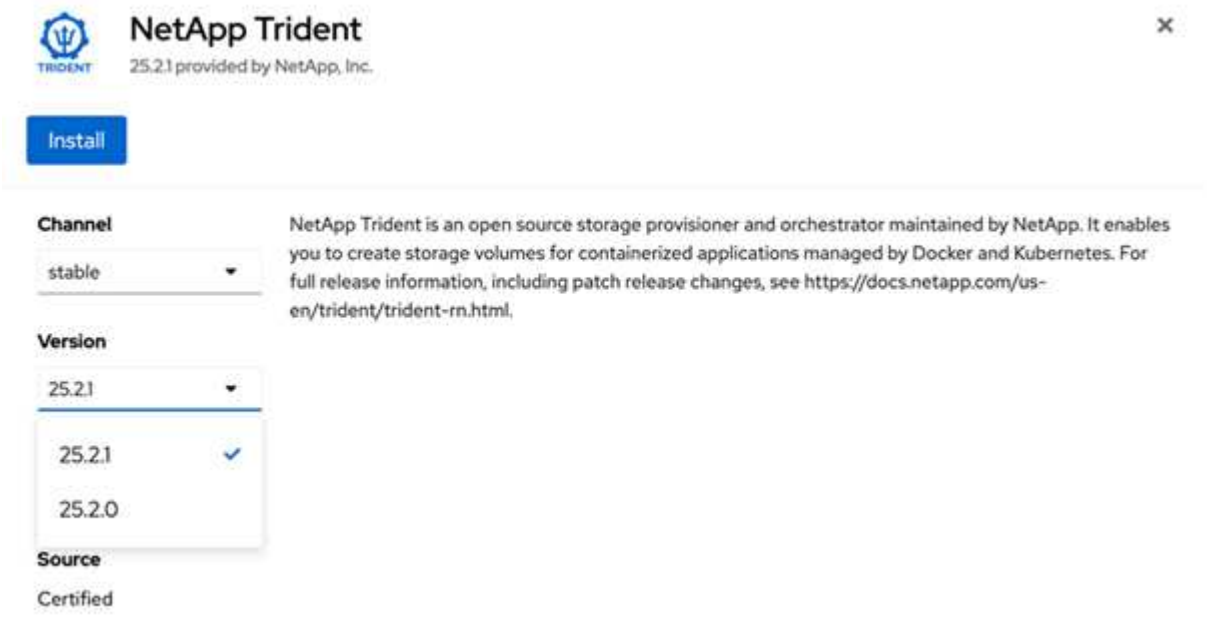
Passos

1. Navegue até OpenShift OperatorHub e pesquise por NetApp Trident.



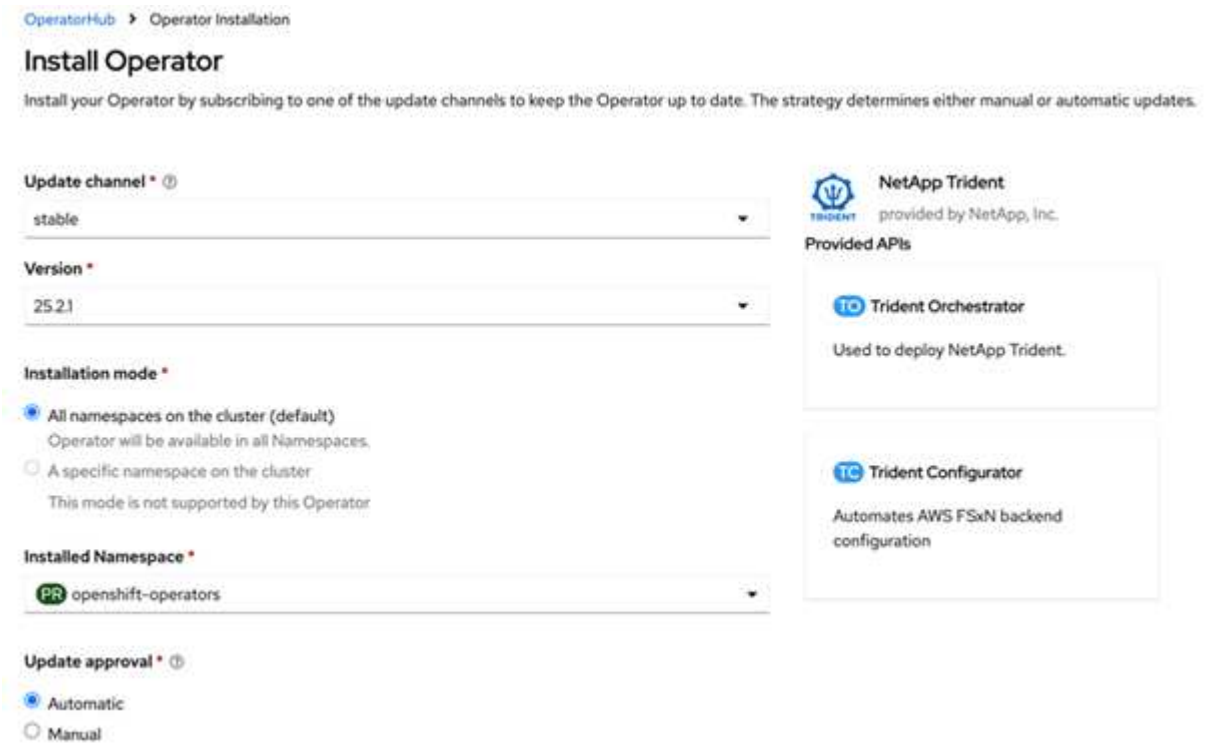
2. Clique em **NetApp Trident** para abrir as configurações de instalação.

3. Selecione as opções necessárias e clique em **Instalar** para abrir a configuração do operador.

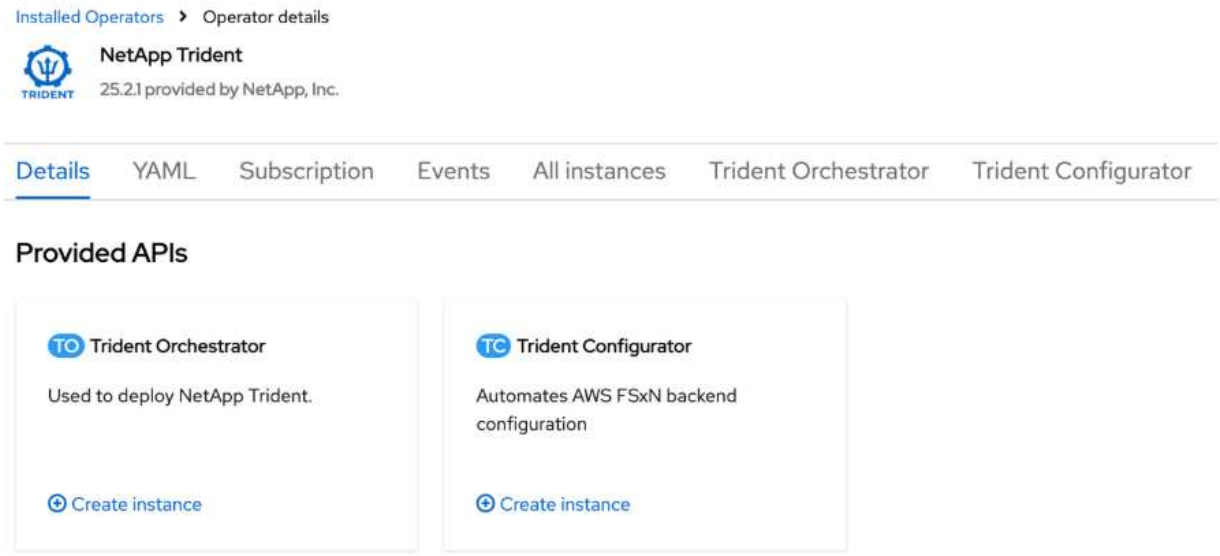


Certifique-se de selecionar a versão mais recente do Operator.

4. Mantenha todos os parâmetros como estão e clique em **Install**.



5. Clique em **View Operator** para visualizar os detalhes do Operator.



6. Clique em **visualização YAML** e cole o seguinte no formulário:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
  namespace: openshift-operators
spec:
  IPv6: false
  debug: false
  nodePrep:
  - iscsi
  imageRegistry: ''
  k8sTimeout: 180s
  namespace: trident
  silenceAutosupport: false
```

A interface do usuário fornece um exemplo padrão. Você pode editá-lo diretamente em vez de copiar uma configuração completa.

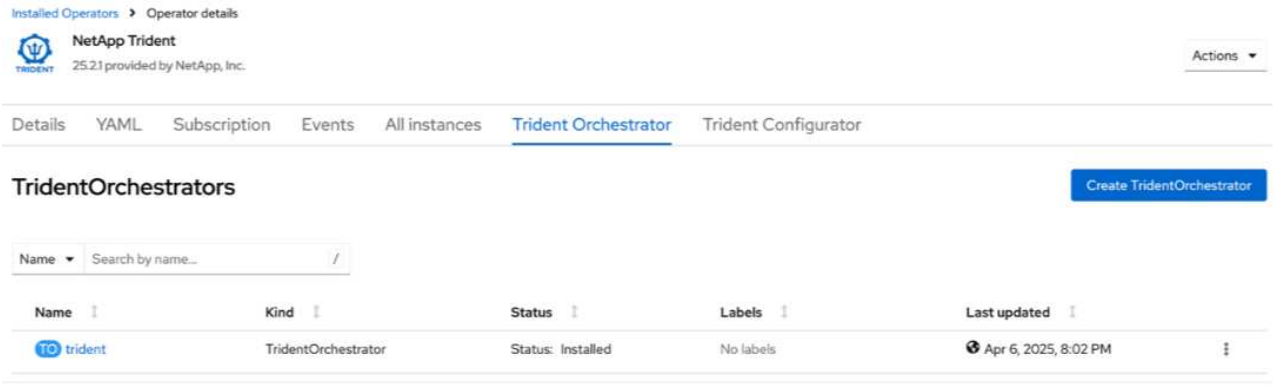
Opcional: ativar concorrência

Para habilitar a concorrência, adicione o seguinte campo à especificação:

```
enableConcurrency: true
```

- Red Hat Enterprise Linux CoreOS (RHCOS) não possui iSCSI habilitado e configurado.
- Você pode adicionar o `nodePrep` parâmetro para configurar e habilitar os serviços iSCSI e Multipath em todos os nós de trabalho OpenShift.
- A partir do OpenShift 4.19, a versão mínima do Trident compatível com este recurso é 25.06.1.

1. Clique em **Criar**; o Trident Orchestrator será totalmente instalado.



Desinstalar o operador Trident

Passos

1. Selecione o operador Trident na lista de operadores instalados.
2. Selecione se deseja excluir todas as instâncias do operando do operador.

Se você não selecionar a caixa de seleção **Excluir todas as instâncias de operandos deste operador**, Trident não será desinstalado.

3. Clique em **Uninstall**.

Mude para o operador Trident certificado OpenShift

Você pode migrar para o operador Trident certificado pela Red Hat OpenShift a partir do operador da comunidade, de uma instalação baseada em Helm ou de um operador implantado manualmente. O processo para cada método envolve desinstalar o operador existente e depois instalar o operador certificado usando o OperatorHub.

Antes de começar

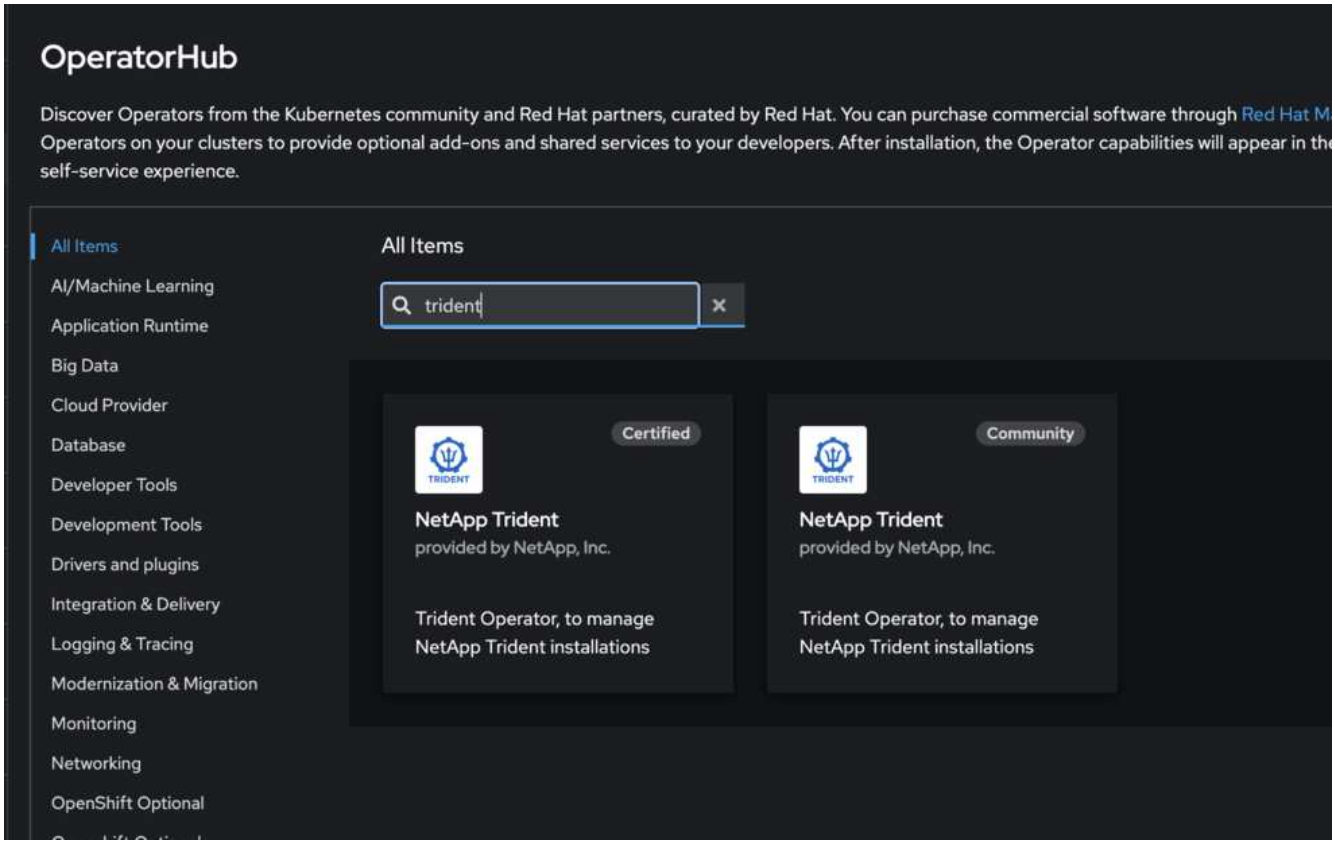
Antes de iniciar a instalação, "prepare seu ambiente para a instalação do Trident".

Não exclua o `TridentOrchestrator` recurso personalizado (CR) durante o processo de desinstalação. O `TridentOrchestrator` CR preserva a configuração do seu backend e da classe de armazenamento, que é necessária após a instalação do operador certificado.

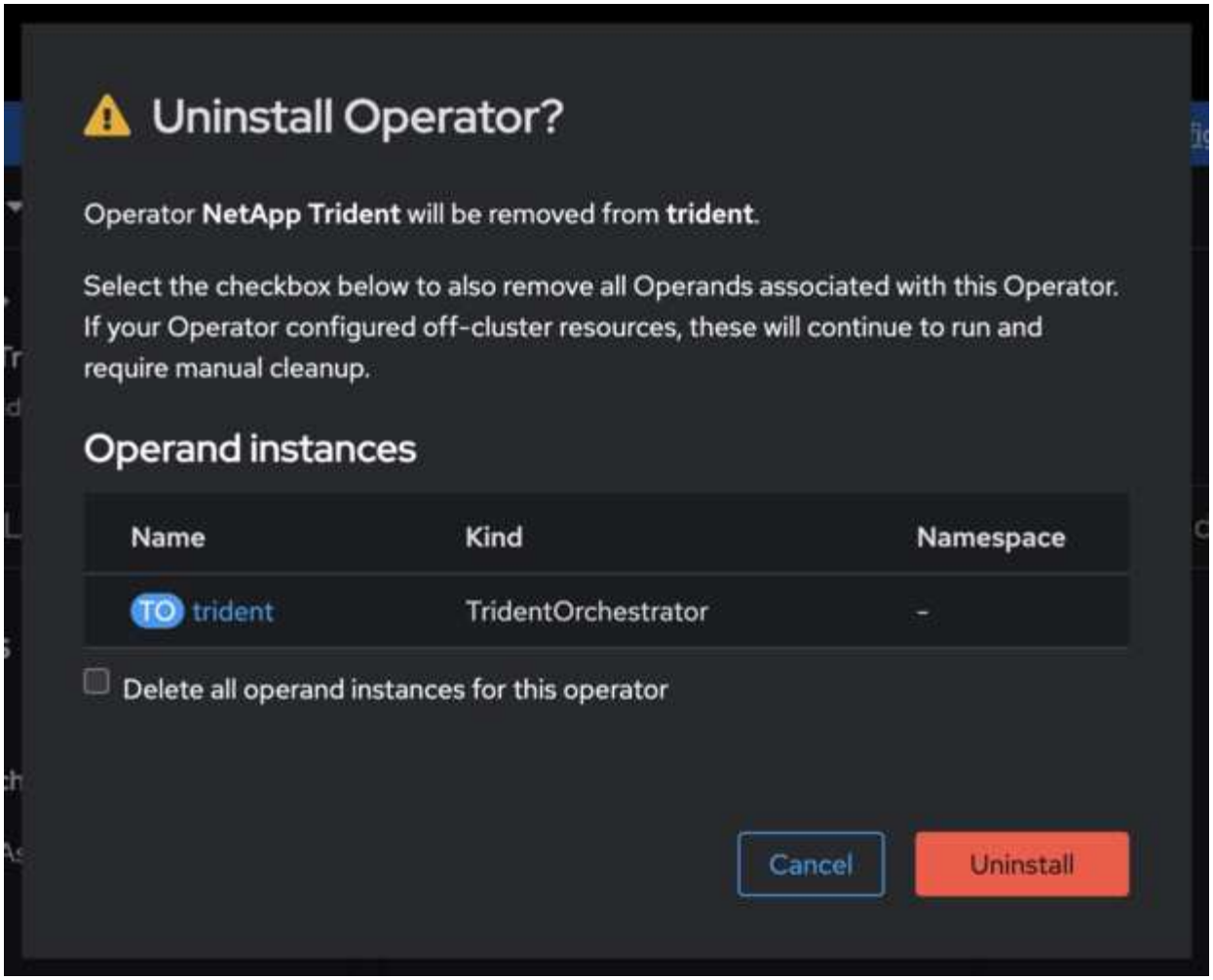
Mude do operador comunitário

Passos

1. Use o console do OpenShift para navegar até o OperatorHub.



2. Encontre o operador da comunidade NetApp Trident.



Não selecione **Excluir todas as instâncias de operandos deste operador**.

3. Clique em **Uninstall**.
4. Após a desinstalação ser concluída, prossiga para Instale o operador certificado OpenShift.

Migrar de uma instalação baseada em operador Helm

Passos

1. Liste o release do Helm para sua instalação do Trident:

```
helm ls -n trident
```

2. Desinstale o release do Helm:

```
helm uninstall <release-name> -n trident
```

3. Após a desinstalação ser concluída, prossiga para Instale o operador certificado OpenShift.

Mudar de um operador implantado manualmente

Se você instalou Trident implantando manualmente o operador usando um `bundle.yaml` do pacote de instalação, remova-o excluindo o mesmo manifesto.

Passos

1. Exclua a implantação do operador usando o bundle manifest:

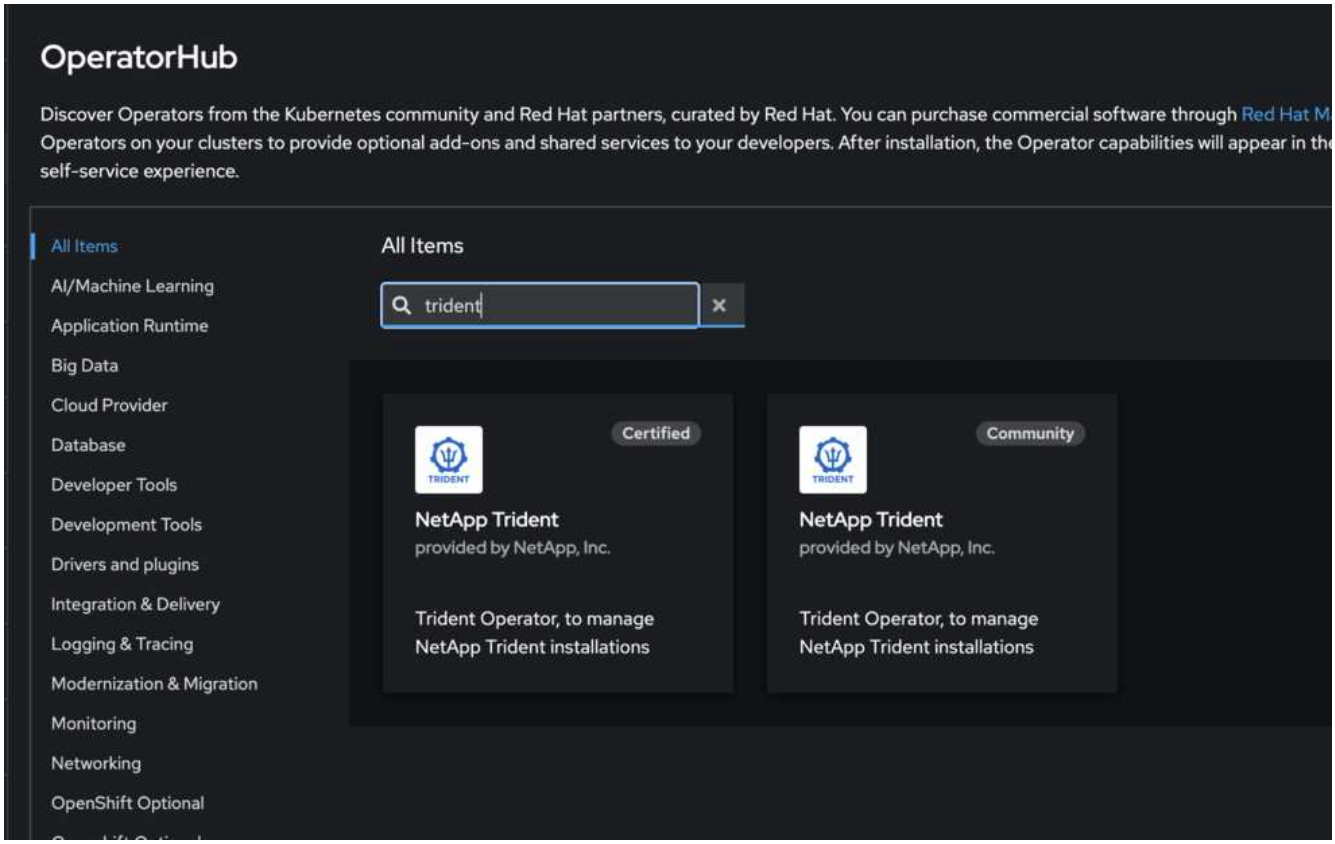
```
kubectl delete -f deploy/bundle.yaml -n trident
```

2. Após a desinstalação ser concluída, prossiga para Instale o operador certificado OpenShift.

Instale o operador certificado OpenShift

Passos

1. Navegue até o Red Hat OperatorHub.
2. Procure e selecione o NetApp Trident Operator.



3. Siga as instruções na tela para instalar o operador.

Verificação

- Verifique o OperatorHub no console para garantir que o novo operador certificado foi instalado com sucesso.

Informações sobre direitos autorais

Informações sobre marcas comerciais