



Instale usando o operador Trident

Trident

NetApp
October 30, 2026

Índice

Instale usando o operador Trident	1
Implante manualmente o operador Trident (modo padrão)	1
Informações críticas sobre Trident 26.06	1
Implante manualmente o operador Trident e instale o Trident	1
Verifique a instalação	4
Implantar manualmente o operador Trident (modo offline)	6
Informações essenciais sobre Trident	6
Implante manualmente o operador Trident e instale o Trident	6
Verifique a instalação	10
Implantar o operador Trident usando Helm (modo padrão)	11
Informações críticas sobre Trident 26.06	11
Implante o operador Trident e instale o Trident usando o Helm	12
Passe os dados de configuração durante a instalação	12
Opções de configuração	13
Implantar o operador Trident usando Helm (modo offline)	19
Informações críticas sobre Trident 26.06	19
Implante o operador Trident e instale o Trident usando o Helm	19
Passe os dados de configuração durante a instalação	20
Opções de configuração	21
Personalizar instalação do Trident	27
Entendendo os pods do controlador e os pods do nó	27
Opções de configuração	27
Configurações de exemplo	31

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 26.06

Você deve ler as seguintes informações críticas sobre Trident.

▼ Informações críticas sobre Trident

- Kubernetes 1.37 agora é compatível com Trident. Atualize Trident antes de atualizar o 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`.

▼ Details

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/v26.06.0/trident-
installer-26.06.0.tar.gz
tar -xf trident-installer-26.06.0.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 replicaset 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:26.06
    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:26.06.0
  Message: Trident installed Namespace:
trident
  Status: Installed
  Version: v26.06.0
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 <code>TridentOrchestrator CR</code> .
Instalado	Trident foi instalado com sucesso.
Desinstalando	O operador está desinstalando Trident porque <code>spec.uninstall=true</code> .
Desinstalado	Trident foi desinstalado.

Status	Descrição
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 |
+-----+-----+
| 26.06.0       | 26.06.0       |
+-----+-----+
```

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 1.37 agora é compatível com Trident. Atualize Trident antes de atualizar o 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`.

▼ Details

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-26.06.0.tar.gz
tar -xf trident-installer-26.06.0.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:26.06.0` 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:26.06.0` se todas as suas imagens estiverem localizadas em um único registro.
- `image: <your-registry>/netapp/trident-operator:26.06.0` 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 replicaset 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:26.06"
tridentImage: "<your-registry>/trident:26.06.0"
```

Imagens em diferentes repositórios

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.06"
tridentImage: "<your-registry>/trident:26.06.0"
```

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:26.06
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.06.0
Status:
  Current Installation Params:
    IPv6:              false
    Autosupport Hostname:
    Autosupport Image:  <your-registry>/trident-autosupport:26.06
    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:26.06.0
Message:             Trident installed
Namespace:           trident
Status:              Installed
Version:             v26.06.0
Events:
  Type Reason Age From Message ----
  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 <code>TridentOrchestrator CR</code> .
Instalado	Trident foi instalado com sucesso.
Desinstalando	O operador está desinstalando Trident porque <code>spec.uninstall=true</code> .
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 <code>TridentOrchestrator</code> 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 1m	6/6	Running	0
trident-node-linux-mr6zc 1m	2/2	Running	0
trident-node-linux-xrp7w 1m	2/2	Running	0
trident-node-linux-zh2jt 1m	2/2	Running	0
trident-operator-766f7b8658-ldzsv 3m	1/1	Running	0

Usando tridentctl

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

```
./tridentctl -n trident version
```

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 26.06.0       | 26.06.0       |
+-----+-----+
```

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 26.06

Você deve ler as seguintes informações críticas sobre Trident.

▼ Informações críticas sobre Trident

- Kubernetes 1.37 agora é compatível com Trident. Atualize Trident antes de atualizar o 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 100.2606.0 é a versão do Trident que você está instalando.

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0  
--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
<code>--values</code> (ou <code>-f</code>)	Especifique um arquivo YAML com sobrescritas. Isso pode ser especificado várias vezes e o arquivo mais à direita terá precedência.
<code>--set</code>	Especifique as substituições na linha de comando.

Por exemplo, para alterar o valor padrão de `debug`, execute o seguinte comando onde 100.2606.0 é a versão do Trident que você está instalando:

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0
```

```
--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
<code>nodeSelector</code>	Rótulos de nós para atribuição de pods	
<code>podAnnotations</code>	Anotações de pod	
<code>deploymentAnnotations</code>	Anotações de deployment	
<code>tolerations</code>	Tolerâncias para atribuição de pod	
<code>affinity</code>	Afinidade para atribuição de pod	<pre>affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux</pre> <div> Não remova a afinidade padrão do arquivo <code>values.yaml</code>. Quando desejar fornecer uma afinidade personalizada, estenda a afinidade padrão.</div>
<code>tridentControllerPluginNodeSelector</code>	Seletores de nós adicionais para pods. Consulte Entendendo os pods do controlador e os pods do nó para obter detalhes.	

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

Opção	Descrição	Padrão
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
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 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.	""

Opção	Descrição	Padrão
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.	""
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-xxxxxxxxxx") 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

Opção	Descrição	Padrão
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.	
enableConcurrency	Permite operações simultâneas do controlador do Trident para melhorar o desempenho. Para obter mais informações sobre os drivers compatíveis, consulte o xref:./trident-install/ Controller Scalability documentation .	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	<p>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.</p> <p>Para mais informações sobre como configurar solicitações e limites de recursos, consulte "Gerenciamento de recursos para pods e contêineres".</p> <div>  <ul style="list-style-type: none"> NÃO altere os nomes de nenhum container ou campo. NÃO altere a indentação - a indentação do YAML é fundamental para a análise correta. </div> <div>  <ul style="list-style-type: none"> Por padrão, não há limites aplicados - apenas as solicitações possuem valores padrão e são aplicadas automaticamente 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. </div>	<pre>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:</pre>

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

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 deseja 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 26.06

Você deve ler as seguintes informações críticas sobre Trident.

▼ Informações críticas sobre Trident

- Kubernetes 1.37 agora é compatível com Trident. Atualize Trident antes de atualizar o 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, `100.2606.0` é a versão do Trident que você está instalando.

Imagens em um registro

```
helm install <name> netapp-trident/trident-operator --version  
100.2606.0 --set imageRegistry=<your-registry> --create-namespace  
--namespace <trident-namespace> --set nodePrep={iscsi}
```

Imagens em diferentes repositórios

```
helm install <name> netapp-trident/trident-operator --version  
100.2606.0 --set imageRegistry=<your-registry> --set operatorImage  
=<your-registry>/trident-operator:26.06.0 --set  
tridentAutosupportImage=<your-registry>/trident-autosupport:26.06  
--set tridentImage=<your-registry>/trident:26.06.0 --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 `100.2606.0` é a versão do Trident que você está instalando:

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0
--create-namespace --namespace trident --set tridentDebug=true
```

Para adicionar o valor `nodePrep`, execute o seguinte comando:

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0
--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
<code>nodeSelector</code>	Rótulos de nós para atribuição de pods	
<code>podAnnotations</code>	Anotações de pod	
<code>deploymentAnnotations</code>	Anotações de deployment	
<code>tolerations</code>	Tolerâncias para atribuição de pod	

Opção	Descrição	Padrão
affinity	Afinidade para atribuição de pod	<pre> affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux </pre> <div>  <p>Não remova a afinidade padrão do arquivo values.yaml. Quando desejar fornecer uma afinidade personalizada, estenda a afinidade padrão.</p> </div>
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
<code>tridentNodePluginNodeSelector</code>	Seletores de nós adicionais para pods. Consulte "Entendendo os pods do controlador e os pods do nó" para obter detalhes.	
<code>tridentNodePluginTolerations</code>	Substitui as tolerâncias do Kubernetes para pods. Consulte "Entendendo os pods do controlador e os pods do nó" para obter detalhes.	
<code>imageRegistry</code>	Identifica o registro para as <code>trident-operator</code> , <code>trident</code> e outras imagens. Deixe em branco para aceitar o padrão. IMPORTANTE: Ao instalar Trident em um repositório privado, se você estiver usando o <code>imageRegistry</code> para especificar a localização do repositório, não use <code>/netapp/</code> no caminho do repositório.	""
<code>imagePullPolicy</code>	Define a política de pull de imagens para o <code>trident-operator</code> .	<code>IfNotPresent</code>
<code>imagePullSecrets</code>	Define os segredos de extração de imagens para as <code>trident-operator</code> , <code>trident</code> e outras imagens.	
<code>kubeletDir</code>	Permite substituir a localização do host do estado interno do kubelet.	<code>"/var/lib/kubelet"</code>
<code>operatorLogLevel</code>	Permite definir o nível de log do operador Trident para: <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code> , ou <code>fatal</code> .	<code>"info"</code>
<code>operatorDebug</code>	Permite definir o nível de registro do operador Trident como depuração.	<code>true</code>
<code>operatorImage</code>	Permite a substituição completa da imagem para <code>trident-operator</code> .	""
<code>operatorImageTag</code>	Permite sobrescrever a tag da <code>trident-operator</code> imagem.	""
<code>tridentIPv6</code>	Permite habilitar Trident para funcionar em clusters IPv6.	<code>false</code>

Opção	Descrição	Padrão
tridentK8sTimeout	<p>Substitui o tempo limite padrão de 180 segundos para a maioria das operações da API do Kubernetes (se diferente de zero, em segundos).</p> <div>  <p>O `tridentK8sTimeout` parâmetro aplica-se apenas à instalação do Trident.</p> </div>	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.	""
tridentImage	Permite a substituição completa da imagem para Trident.	""

Opção	Descrição	Padrão
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 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	<p>Permite que Trident prepare os nós do cluster Kubernetes para gerenciar volumes usando o protocolo de storage de dados especificado. Atualmente, <code>iscsi</code> é o único valor suportado.</p> <div>  <p>A partir do OpenShift 4.19, a versão mínima do Trident compatível com este recurso é 25.06.1.</p> </div>	

Opção	Descrição	Padrão
resources	<p>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.</p> <p>Para mais informações sobre como configurar solicitações e limites de recursos, consulte "Gerenciamento de recursos para pods e contêineres".</p> <div>  <ul style="list-style-type: none"> NÃO altere os nomes de nenhum container ou campo. NÃO altere a indentação - a indentação do YAML é fundamental para a análise correta. </div> <div>  <ul style="list-style-type: none"> 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 <code>status.CurrentInstallati</code> </div>	<pre>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:</pre>

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 deseja 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



`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
<code>namespace</code>	Namespace para instalar Trident	"default"
<code>debug</code>	Ative a depuração para Trident	false
<code>enableForceDetach</code>	<code>ontap-san</code> , <code>ontap-san-economy</code> , <code>ontap-nas</code> , e <code>ontap-nas-economy</code> 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 informações, consulte " Automatizando o failover de aplicações stateful com Trident ".	false
<code>windows</code>	Definir como <code>true</code> permite instalação em nós de trabalho Windows.	false

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	<p>Tempo limite para operações do Kubernetes.</p> <div>  <p>O `k8sTimeout` parâmetro aplica-se apenas à instalação do Trident.</p> </div>	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:26.06"
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	
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

Parâmetro	Descrição	Padrão
controllerPluginNodeSelector	Seletores de nós adicionais para pods. Segue o mesmo formato que <code>pod.spec.nodeSelector</code> .	Sem valor padrão; opcional
controllerPluginTolerations	Substitui as tolerâncias do Kubernetes para pods. Segue o mesmo formato que <code>pod.spec.Tolerations</code> .	Sem valor padrão; opcional
nodePluginNodeSelector	Seletores de nós adicionais para pods. Segue o mesmo formato que <code>pod.spec.nodeSelector</code> .	Sem valor padrão; opcional
nodePluginTolerations	Substitui as tolerâncias do Kubernetes para pods. Segue o mesmo formato que <code>pod.spec.Tolerations</code> .	Sem valor padrão; opcional
nodePrep	<p>Permite que Trident prepare os nós do cluster Kubernetes para gerenciar volumes usando o protocolo de storage de dados especificado.</p> <p>Atualmente, <code>iscsi</code> é o único valor suportado.</p> <div>  <p>A partir do OpenShift 4.19, a versão mínima do Trident compatível com este recurso é 25.06.1.</p> </div>	
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 do Trident para melhorar o desempenho. Para obter mais informações sobre os drivers compatíveis, consulte o xref:./trident-install/ Controller Scalability documentation .	resources

Parâmetro	Descrição	Padrão
<p>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.</p> <p>Para mais informações sobre como configurar solicitações e limites de recursos, consulte "Gerenciamento de recursos para pods e contêineres".</p> <div>  <ul style="list-style-type: none"> NÃO altere os nomes de nenhum container ou campo. NÃO altere a indentação - a indentação do YAML é fundamental para a análise correta. </div> <div>  <ul style="list-style-type: none"> Por padrão, não há limites aplicados - apenas as solicitações possui </div>	<pre> 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: </pre>	httpsMetrics

Parâmetro	Descrição	Padrão
Habilite o HTTPS para o endpoint de métricas do Prometheus.	falso	hostNetwork



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-xxxxxxxxxxxx'
```

▼ 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
```

Informações sobre direitos autorais

Copyright © 2026 NetApp, Inc. Todos os direitos reservados. Impresso nos EUA. Nenhuma parte deste documento protegida por direitos autorais pode ser reproduzida de qualquer forma ou por qualquer meio — gráfico, eletrônico ou mecânico, incluindo fotocópia, gravação, gravação em fita ou storage em um sistema de recuperação eletrônica — sem permissão prévia, por escrito, do proprietário dos direitos autorais.

O software derivado do material da NetApp protegido por direitos autorais está sujeito à seguinte licença e isenção de responsabilidade:

ESTE SOFTWARE É FORNECIDO PELA NETAPP "NO PRESENTE ESTADO" E SEM QUAISQUER GARANTIAS EXPRESSAS OU IMPLÍCITAS, INCLUINDO, SEM LIMITAÇÕES, GARANTIAS IMPLÍCITAS DE COMERCIALIZAÇÃO E ADEQUAÇÃO A UM DETERMINADO PROPÓSITO, CONFORME A ISENÇÃO DE RESPONSABILIDADE DESTES DOCUMENTOS. EM HIPÓTESE ALGUMA A NETAPP SERÁ RESPONSÁVEL POR QUALQUER DANO DIRETO, INDIRETO, INCIDENTAL, ESPECIAL, EXEMPLAR OU CONSEQUENCIAL (INCLUINDO, SEM LIMITAÇÕES, AQUISIÇÃO DE PRODUTOS OU SERVIÇOS SOBRESSALENTE; PERDA DE USO, DADOS OU LUCROS; OU INTERRUPÇÃO DOS NEGÓCIOS), INDEPENDENTEMENTE DA CAUSA E DO PRINCÍPIO DE RESPONSABILIDADE, SEJA EM CONTRATO, POR RESPONSABILIDADE OBJETIVA OU PREJUÍZO (INCLUINDO NEGLIGÊNCIA OU DE OUTRO MODO), RESULTANTE DO USO DESTES SOFTWARES, MESMO SE ADVERTIDA DA RESPONSABILIDADE DE TAL DANO.

A NetApp reserva-se o direito de alterar quaisquer produtos descritos neste documento, a qualquer momento e sem aviso. A NetApp não assume nenhuma responsabilidade nem obrigação decorrentes do uso dos produtos descritos neste documento, exceto conforme expressamente acordado por escrito pela NetApp. O uso ou a compra deste produto não representam uma licença sob quaisquer direitos de patente, direitos de marca comercial ou quaisquer outros direitos de propriedade intelectual da NetApp.

O produto descrito neste manual pode estar protegido por uma ou mais patentes dos EUA, patentes estrangeiras ou pedidos pendentes.

LEGENDA DE DIREITOS LIMITADOS: o uso, a duplicação ou a divulgação pelo governo estão sujeitos a restrições conforme estabelecido no subparágrafo (b)(3) dos Direitos em Dados Técnicos - Itens Não Comerciais no DFARS 252.227-7013 (fevereiro de 2014) e no FAR 52.227- 19 (dezembro de 2007).

Os dados aqui contidos pertencem a um produto comercial e/ou serviço comercial (conforme definido no FAR 2.101) e são de propriedade da NetApp, Inc. Todos os dados técnicos e software de computador da NetApp fornecidos sob este Contrato são de natureza comercial e desenvolvidos exclusivamente com despesas privadas. O Governo dos EUA tem uma licença mundial limitada, irrevogável, não exclusiva, intransferível e não sublicenciável para usar os Dados que estão relacionados apenas com o suporte e para cumprir os contratos governamentais desse país que determinam o fornecimento de tais Dados. Salvo disposição em contrário no presente documento, não é permitido usar, divulgar, reproduzir, modificar, executar ou exibir os dados sem a aprovação prévia por escrito da NetApp, Inc. Os direitos de licença pertencentes ao governo dos Estados Unidos para o Departamento de Defesa estão limitados aos direitos identificados na cláusula 252.227-7015(b) (fevereiro de 2014) do DFARS.

Informações sobre marcas comerciais

NETAPP, o logotipo NETAPP e as marcas listadas em <http://www.netapp.com/TM> são marcas comerciais da NetApp, Inc. Outros nomes de produtos e empresas podem ser marcas comerciais de seus respectivos proprietários.