Comece agora

Astra Trident

NetApp
January 31, 2025

Índice

Comece agora

Experimente

O NetApp fornece uma imagem de laboratório pronta para uso que pode ser solicitada através "Unidade de teste do NetApp"do .

Saiba mais sobre o Test Drive

O Test Drive oferece um ambiente sandbox que inclui um cluster de Kubernetes de três nós e o Astra Trident instalado e configurado. É uma ótima maneira de se familiarizar com o Astra Trident e explorar seus recursos.

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

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

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

Requisitos

Antes de instalar o Astra Trident, você deve analisar esses requisitos gerais de sistema. Backends específicos podem ter requisitos adicionais.

Informações críticas sobre o Astra Trident 23,01

Você deve ler as seguintes informações críticas sobre o Astra Trident.

** informações essenciais sobre o Astra Trident **

- O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Trident antes de atualizar o Kubernetes.
- O Astra Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Frontens suportados (orquestradores)

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

- Anthos On-Prem (VMware) e Anthos em bare metal 1,12
- Kubernetes 1,21 - 1,27

- Mecanismo do Kubernetes do Mirantis 3,5
- OpenShift 4,9 - 4,12

O operador Trident é suportado com estas versões:

- Anthos On-Prem (VMware) e Anthos em bare metal 1,12
- Kubernetes 1,21 - 1,27
- OpenShift 4,9 - 4,12

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

Antes de atualizar um cluster do Kubernetes do 1,24 para o 1,25 ou posterior que tenha o Astra Trident instalado, "Atualize uma instalação de operador baseada em Helm"consulte .

Backends suportados (armazenamento)

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

- Amazon FSX para NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Cloud Volumes Service para GCP
- FAS/AFF/Selecione 9,5 ou posterior
- NetApp All SAN Array (ASA)
- Software NetApp HCI/Element 11 ou superior

Requisitos de recursos

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

Recurso	Versão do Kubernetes	É necessário ter portões?
CSI Trident	1,21 - 1,27	Não
Instantâneos de volume	1,21 - 1,27	Não
PVC a partir de instantâneos de volume	1,21 - 1,27	Não
Redimensionamento iSCSI PV	1,21 - 1,27	Não
ONTAP bidirectional CHAP	1,21 - 1,27	Não
Políticas de exportação dinâmica	1,21 - 1,27	Não

Recurso	Versão do Kubernetes	É necessário ter portões?
Operador Trident	1,21 - 1,27	Não
Topologia de CSI	1,21 - 1,27	Não

Sistemas operacionais de host testados

Embora o Astra Trident não seja oficialmente compatível com sistemas operacionais específicos, sabe-se que os seguintes itens funcionam:

- Versões do RedHat CoreOS (RHCOS) suportadas pela OpenShift Container Platform (AMD64 e ARM64)
- RHEL 8 OU SUPERIOR (AMD64 E ARM64)
- Ubuntu 22,04 ou posterior (AMD64 e ARM64)
- Windows Server 2019 (AMD64 bits)

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

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

Configuração de host

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

"Prepare o nó de trabalho"

Configuração do sistema de storage

O Astra Trident pode exigir alterações em um sistema de storage antes que uma configuração de back-end o use.

"Configurar backends"

Portas Astra Trident

O Astra Trident requer acesso a portas específicas para comunicação.

"Portas Astra Trident"

Imagens de contêineres e versões correspondentes do Kubernetes

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

Versão do Kubernetes	Imagem do recipiente
v1.21.0	• docker.io/NetApp/Trident:23.04.0 • docker.io/NetApp/Trident-AutoSupport:23,04 • provisionador do registry.k8s.io/sig-storage/csi:v3,4.1 • registry.k8s.io/sig-storage/csi-attacher:v4,2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6,2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2.7.0 • docker.io/NetApp/Trident-operador:23.04.0 (opcional)
v1.22.0	• docker.io/NetApp/Trident:23.04.0 • docker.io/NetApp/Trident-AutoSupport:23,04 • provisionador do registry.k8s.io/sig-storage/csi:v3,4.1 • registry.k8s.io/sig-storage/csi-attacher:v4,2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6,2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2.7.0 • docker.io/NetApp/Trident-operador:23.04.0 (opcional)
v1.23.0	• docker.io/NetApp/Trident:23.04.0 • docker.io/NetApp/Trident-AutoSupport:23,04 • provisionador do registry.k8s.io/sig-storage/csi:v3,4.1 • registry.k8s.io/sig-storage/csi-attacher:v4,2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6,2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2.7.0 • docker.io/NetApp/Trident-operador:23.04.0 (opcional)

Versão do Kubernetes	Imagem do recipiente
v1.24.0	• docker.io/NetApp/Trident:23.04.0 • docker.io/NetApp/Trident-AutoSupport:23,04 • provisionador do registry.k8s.io/sig-storage/csi:v3,4.1 • registry.k8s.io/sig-storage/csi-attacher:v4,2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6,2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2.7.0 • docker.io/NetApp/Trident-operador:23.04.0 (opcional)
v1.25.0	• docker.io/NetApp/Trident:23.04.0 • docker.io/NetApp/Trident-AutoSupport:23,04 • provisionador do registry.k8s.io/sig-storage/csi:v3,4.1 • registry.k8s.io/sig-storage/csi-attacher:v4,2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6,2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2.7.0 • docker.io/NetApp/Trident-operador:23.04.0 (opcional)
v1.26.0	• docker.io/NetApp/Trident:23.04.0 • docker.io/NetApp/Trident-AutoSupport:23,04 • provisionador do registry.k8s.io/sig-storage/csi:v3,4.1 • registry.k8s.io/sig-storage/csi-attacher:v4,2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6,2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2.7.0 • docker.io/NetApp/Trident-operador:23.04.0 (opcional)

Versão do Kubernetes	Imagem do recipiente
v1.27.0	• docker.io/NetApp/Trident:23.04.0 • docker.io/NetApp/Trident-AutoSupport:23,04 • provisionador do registry.k8s.io/sig-storage/csi:v3,4.1 • registry.k8s.io/sig-storage/csi-attacher:v4,2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6,2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2.7.0 • docker.io/NetApp/Trident-operador:23.04.0 (opcional)

No Kubernetes versão 1,21 e posterior, use a imagem validada `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v6.x` somente se a `v1` versão estiver servindo o `volumesnapshots.snapshot.storage.k8s.gcr.io` CRD. Se a `v1beta1` versão estiver servindo o CRD com/sem a `v1` versão, use a imagem validada `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v3.x`.

Instale o Astra Trident

Saiba mais sobre a instalação do Astra Trident

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

Informações críticas sobre o Astra Trident 23,04

Você deve ler as seguintes informações críticas sobre o Astra Trident.

informações essenciais sobre o Astra Trident

- O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Trident antes de atualizar o Kubernetes.
- O Astra Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Antes de começar

Independentemente do seu caminho de instalação, você deve ter:

- Privileges completo para um cluster Kubernetes compatível com execução de uma versão compatível do Kubernetes e requisitos de recursos habilitados. Reveja a "requisitos" para obter detalhes.
- Acesso a um sistema de storage NetApp compatível.
- Funcionalidade de montar volumes de todos os nós de trabalho do Kubernetes.
- Um host Linux com `kubectl` (ou `oc`, se você estiver usando o OpenShift) instalado e configurado para gerenciar o cluster do Kubernetes que deseja usar.
- A `KUBECONFIG` variável de ambiente configurada para apontar para a configuração do cluster do Kubernetes.
- Se você estiver usando o Kubernetes com Docker Enterprise "Siga os passos para ativar o acesso CLI", .

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

Escolha o método de instalação

Selecione o método de instalação correto para você. Você também deve rever as considerações "movendo-se entre métodos"antes de tomar sua decisão.

Utilizando 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 Astra Trident. Você pode até mesmo "Personalize a implantação do seu operador Trident" usar os atributos no `TridentOrchestrator` recurso personalizado (CR).

Os benefícios de usar o operador Trident incluem:

Astra Trident Object creation

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

- ServiceAccount para o operador
- ClusterRole e ClusterRoleBinding para o ServiceAccount
- PodSecurityPolicy dedicada (para Kubernetes 1,25 e versões anteriores)
- O próprio operador

** capacidade de autorrecuperação **

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

** atualizações fáceis para existente**

Você pode facilmente atualizar uma implantação existente com o operador. Você só precisa editar o `TridentOrchestrator` CR para fazer atualizações em uma instalação.

Por exemplo, considere um cenário em que você precisa habilitar o Astra Trident para gerar logs de depuração. Para fazer isso, corrija o `TridentOrchestrator` para definir `spec.debug` como `true`:

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

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

** handling de atualização automática do Kubernetes**

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

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

** gerenciamento de clusters do Kubernetes usando o BlueXP (anteriormente conhecido como Cloud Manager) **

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

Utilização `tridentctl`

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

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

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

Escolha o modo de instalação

Determine seu processo de implantação com base no *modo de instalação* (padrão, Offline ou remoto) exigido pela sua organização.

Instalação padrão

Essa é a maneira mais fácil de instalar o Astra Trident e funciona para a maioria dos ambientes que não impõem restrições de rede. O modo de instalação padrão usa Registros padrão para armazenar (`registry.k8s.io`imagens Trident (`docker.io`) e CSI necessárias.

Quando você usa o modo padrão, o instalador do Astra Trident:

- Obtém as imagens de contentor pela Internet
- Cria um daemonset de implantação ou nó, que ativa pods do Astra Trident em todos os nós qualificados no cluster do Kubernetes

Instalação offline

O modo de instalação off-line pode ser necessário em um local seguro ou protegido. Nesse cenário, você pode criar um único Registro privado espelhado ou dois Registros espelhados para armazenar imagens Trident e CSI necessárias.

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

Selecione o processo com base no seu método e modo

Depois de tomar suas decisões, selecione o processo apropriado.

Método	Modo de instalação
Operador Trident (manualmente)	"Instalação padrão" "Instalação offline"
Operador Trident (Helm)	"Instalação padrão" "Instalação offline"
`tridentctl`	"Instalação padrão ou offline"

Movendo-se entre os métodos de instalação

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

- Sempre use o mesmo método para instalar e desinstalar o Astra Trident. Se você implantou com `tridentctl`o , use a versão apropriada `tridentctl` do binário para desinstalar o Astra Trident. Da mesma forma, se você estiver implantando com o operador, edite o `TridentOrchestrator` CR e configure `spec.uninstall=true` para desinstalar o Astra Trident.
- Se você tiver uma implantação baseada em operador que deseja remover e usar `tridentctl` para implantar o Astra Trident, primeiro edite `TridentOrchestrator` e configure `spec.uninstall=true` para desinstalar o Astra Trident. Em seguida, exclua `TridentOrchestrator` e a implantação do operador. Você pode instalar usando `tridentctl`o .
- Se você tiver uma implantação manual baseada no operador e quiser usar a implantação do operador Trident baseada no Helm, desinstale manualmente o operador primeiro e execute a instalação do Helm. Isso permite que o Helm implante o operador Trident com as etiquetas e anotações necessárias. Se você não fizer isso, sua implantação de operador Trident baseada em Helm falhará com erro de validação de rótulo e erro de validação de anotação. Se você tem uma `tridentctl` implantação baseada em -, você pode usar a implantação baseada em Helm sem problemas.

Outras opções de configuração conhecidas

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

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

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

Instale usando o operador Trident

Implantar manualmente o operador Trident (modo padrão)

Você pode implantar manualmente o operador Trident para instalar o Astra Trident. Este processo aplica-se a instalações onde as imagens de contentor exigidas pelo Astra Trident não são armazenadas em um Registro privado. Se tiver um registo de imagens

privado, utilize o "processo para implantação off-line".

Informações críticas sobre o Astra Trident 23,04

Você deve ler as seguintes informações críticas sobre o Astra Trident.

** informações essenciais sobre o Astra Trident **

- O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Trident antes de atualizar o Kubernetes.
- O Astra Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Implante manualmente o operador Trident e instale o Trident

Revise "a 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 trabalho e "Cluster compatível com Kubernetes"se você tem o Privileges necessário.

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

1. Verifique sua versão do Kubernetes:

```
kubectl version
```

2. Verifique o Privileges do administrador do cluster:

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

3. Verifique se você pode iniciar um pod que usa uma imagem do Docker Hub e alcançar seu sistema de armazenamento pela rede de pod:

```
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 Astra Trident contém tudo o que você precisa para implantar o operador Trident e instalar o Astra Trident. Baixe e extraia a versão mais recente do instalador do Trident a partir "A seção *assets* no GitHub"do .

```
wget https://github.com/NetApp/trident/releases/download/v23.04.0/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
```

Passo 2: Crie o `TridentOrchestrator` CRD

Crie a `TridentOrchestrator` Definição de recurso personalizada (CRD). Você criará um `TridentOrchestrator` recurso personalizado mais tarde. Use a versão apropriada do CRD YAML em `deploy/crds` para criar o `TridentOrchestrator` CRD.

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

Etapa 3: Implante o operador Trident

O instalador do Astra 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 o Astra Trident usando uma configuração padrão.

- Para clusters que executam o Kubernetes 1,24 ou anterior, `bundle_pre_1_25.yaml` use o .
- Para clusters que executam o Kubernetes 1,25 ou posterior, `bundle_post_1_25.yaml` use o .

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, atualizar `serviceaccount.yaml` `clusterrolebinding.yaml` `operator.yaml` e gerar o arquivo do pacote usando o `kustomization.yaml`.

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

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

  b. Compile o pacote usando o seguinte comando onde *<bundle>* está `bundle_pre_1_25` ou

`bundle_post_1_25` baseado na sua versão do Kubernetes.

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

Passos

1. Crie os recursos e implante o operador:

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

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

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

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

Passo 4: Crie o `TridentOrchestrator` e instale o Trident

Agora você pode criar e instalar o `TridentOrchestrator` Astra Trident. Opcionalmente, você pode "Personalize a instalação do Trident" usar os atributos na `TridentOrchestrator` especificação.

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

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:23.04
    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:23.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v23.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Verifique a instalação

Existem várias maneiras de verificar sua instalação.

`TridentOrchestrator`Usando o status

O status de `TridentOrchestrator` indica se a instalação foi bem-sucedida e exibe a versão do Trident

instalado. Durante a instalação, o status das `TridentOrchestrator` alterações de `Installing` para `Installed`. Se você observar o `Failed` status e o operador não conseguir recuperar sozinho, "verifique os logs".

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

Usando o status de criação do pod

Você pode confirmar se a instalação do Astra Trident foi concluída analisando 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
```

Utilização `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.04.0        | 23.04.0        |
+----------------+----------------+
```

O que vem a seguir

Agora você pode "crie um back-end e uma classe de storage, provisione um volume e monte o volume em um pod".

Implantar manualmente o operador Trident (modo off-line)

Você pode implantar manualmente o operador Trident para instalar o Astra Trident. Este processo aplica-se a instalações onde as imagens de contentor exigidas pelo Astra Trident são armazenadas em um Registro privado. Se não tiver um registo de imagens privado, utilize o "processo para implantação padrão".

Informações críticas sobre o Astra Trident 23,04

Você deve ler as seguintes informações críticas sobre o Astra Trident.

informações essenciais sobre o Astra Trident

- O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Trident antes de atualizar o Kubernetes.
- O Astra Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Implante manualmente o operador Trident e instale o Trident

Revise "a 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 trabalho e "Cluster compatível com Kubernetes"que você tem o Privileges necessário.

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

1. Verifique sua versão do Kubernetes:

```
kubectl version
```

2. Verifique o Privileges do administrador do cluster:

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

3. Verifique se você pode iniciar um pod que usa uma imagem do Docker Hub e alcançar seu sistema de armazenamento pela rede de pod:

```
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 Astra Trident contém tudo o que você precisa para implantar o operador Trident e instalar o Astra Trident. Baixe e extraia a versão mais recente do instalador do Trident a partir "A seção *assets* no GitHub"do .

```
wget https://github.com/NetApp/trident/releases/download/v23.04.0/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
```

Passo 2: Crie o `TridentOrchestrator` CRD

Crie a `TridentOrchestrator` Definição de recurso personalizada (CRD). Você criará um `TridentOrchestrator` recurso personalizado mais tarde. Use a versão apropriada do CRD YAML em `deploy/crds` para criar o `TridentOrchestrator` CRD:

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

Etapa 3: Atualize a localização do Registro no operador

No `/deploy/operator.yaml`, atualize `image: docker.io/netapp/trident-operator:23.04.0` para refletir a localização do registo de imagens. O "Imagens de Trident e CSI" pode estar localizado em um Registro ou Registros diferentes, mas todas as imagens CSI devem estar localizadas no mesmo Registro. Por exemplo:

- `image: <your-registry>/trident-operator:23.04.0` se todas as suas imagens estiverem localizadas em um registro.

- `image: <your-registry>/netapp/trident-operator:23.04.0` Se a sua imagem Trident estiver localizada num registo diferente das suas imagens CSI.

Etapa 4: Implante o operador Trident

O instalador do Astra 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 o Astra Trident usando uma configuração padrão.

- Para clusters que executam o Kubernetes 1,24 ou anterior, `bundle_pre_1_25.yaml` use o .
- Para clusters que executam o Kubernetes 1,25 ou posterior, `bundle_post_1_25.yaml` use o .

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, atualizar `serviceaccount.yaml` `clusterrolebinding.yaml` `operator.yaml` e gerar o arquivo do pacote usando o `kustomization.yaml`.

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

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

  b. Compile o pacote usando o seguinte comando onde *<bundle>* está `bundle_pre_1_25` ou `bundle_post_1_25` baseado na sua versão do Kubernetes.

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

Passos

1. Crie os recursos e implante o operador:

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

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

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

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

Passo 5: Atualize a localização do registo de imagens no `TridentOrchestrator`

O "Imagens de Trident e CSI" pode estar localizado em um Registro ou Registros diferentes, mas todas as imagens CSI devem estar localizadas no mesmo Registro. Atualize `deploy/crds/tridentorchestrator_cr.yaml` para adicionar as especificações de localização adicionais com base na configuração do seu registo.

Imagens em um Registro

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:23.04"
tridentImage: "<your-registry>/trident:23.04.0"
```

Imagens em diferentes registos

Você deve anexar `sig-storage` ao `imageRegistry` para usar diferentes locais de Registro.

```
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:23.04"
tridentImage: "<your-registry>/netapp/trident:23.04.0"
```

Passo 6: Crie o `TridentOrchestrator` e instale o Trident

Agora você pode criar e instalar o `TridentOrchestrator` Astra Trident. Opcionalmente, você pode usar ainda mais "Personalize a instalação do Trident"os atributos na `TridentOrchestrator` especificação. O exemplo a seguir mostra uma instalação onde as imagens Trident e CSI estão localizadas em diferentes Registros.

```
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>/netapp/trident-autosupport:23.04
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:23.04.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/netapp/trident-
autosupport:23.04
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>/sig-storage
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/netapp/trident:23.04.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v23.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Verifique a instalação

Existem várias maneiras de verificar sua instalação.

`TridentOrchestrator`Usando o status

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

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

Usando o status de criação do pod

Você pode confirmar se a instalação do Astra Trident foi concluída analisando 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
```

Utilização `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.04.0        | 23.04.0        |
+----------------+----------------+
```

O que vem a seguir

Agora você pode "crie um back-end e uma classe de storage, provisione um volume e monte o volume em um pod".

Implantar operador Trident usando Helm (modo padrão)

Você pode implantar o operador Trident e instalar o Astra Trident usando o Helm. Este processo aplica-se a instalações onde as imagens de contentor exigidas pelo Astra Trident não são armazenadas em um Registro privado. Se tiver um registo de imagens privado, utilize o "processo para implantação off-line".

Informações críticas sobre o Astra Trident 23,04

Você deve ler as seguintes informações críticas sobre o Astra Trident.

** informações essenciais sobre o Astra Trident **

- O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Trident antes de atualizar o Kubernetes.
- O Astra Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

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

Usando o Trident"Carta do leme", você pode implantar o operador Trident e instalar o Trident em uma etapa.

Revise "a 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 implantação" que você precisa"Helm versão 3".

Passos

1. Adicione o repositório Astra Trident Helm:

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

2. Use `helm install` e especifique um nome para sua implantação, como no exemplo a seguir, onde `23.04.0` está a versão do Astra Trident que você está instalando.

```
helm install <name> netapp-trident/trident-operator --version 23.04.0
--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 de instalação, como nome, namespace, gráfico, status, versão do aplicativo e número de revisão.

Passe os dados de configuração durante a instalação

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

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

Por exemplo, para alterar o valor padrão `debug` do , execute o seguinte `--set` comando onde `23.04.0` está a versão do Astra Trident que você está instalando:

```
helm install <name> netapp-trident/trident-operator --version 23.04.0
--create-namespace --namespace --set tridentDebug=true
```

Opções de configuração

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

Opção	Descrição	Padrão
`nodeSelector`	Etiquetas de nó para atribuição de pod	
`podAnnotations`	Anotações do pod	

Opção	Descrição	Padrão
`deploymentAnnotations`	Anotações de implantação	
`tolerations`	Tolerâncias para atribuição de pod	
`affinity`	Afinidade para atribuição de pod	
`tridentControllerPluginNodeSelector`	Seletores de nós adicionais para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`tridentControllerPluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`tridentNodePluginNodeSelector`	Seletores de nós adicionais para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`tridentNodePluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`imageRegistry`	Identifica o registo para as `trident-operator`, `trident`e outras imagens. Deixe vazio para aceitar o padrão.	""
`imagePullPolicy`	Define a política de recebimento de imagens para o `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Define os segredos de extração da imagem 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 que o nível de log do operador Trident seja definido como: `trace`, , `debug`, `info` `warn`, , `error` Ou `fatal`.	`"info"`
`operatorDebug`	Permite que o nível de log do operador Trident seja definido como debug.	`true`

Opção	Descrição	Padrão
`operatorImage`	Permite a substituição completa da imagem para `trident-operator`.	""
`operatorImageTag`	Permite substituir a etiqueta da `trident-operator` imagem.	""
`tridentIPv6`	Permite que o Astra Trident funcione em IPv6 clusters.	`false`
`tridentK8sTimeout`	Substitui o tempo limite padrão de 30 segundos para a maioria das operações da API do Kubernetes (se não for zero, em segundos).	`0`
`tridentHttpRequestTimeout`	Substitui o tempo limite padrão de 90 segundos para as solicitações HTTP, `0s` sendo uma duração infinita para o tempo limite. Valores negativos não são permitidos.	`"90s"`
`tridentSilenceAutosupport`	Permite desativar os relatórios periódicos do Astra Trident AutoSupport.	`false`
`tridentAutosupportImageTag`	Permite substituir a tag da imagem para o contentor Astra Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Permite que o Astra Trident AutoSupport Container ligue para casa por meio de um proxy HTTP.	""
`tridentLogFormat`	Define o formato de registo Astra Trident (`text`ou `json`).	`"text"`
`tridentDisableAuditLog`	Desativa o registrador de auditoria Astra Trident.	`true`
`tridentLogLevel`	Permite que o nível de log do Astra Trident seja definido como: `trace`, `debug`, , `info` `warn` , , `error` `fatal` Ou .	`"info"`
`tridentDebug`	Permite que o nível de log do Astra Trident seja definido como `debug`.	`false`
`tridentLogWorkflows`	Permite que fluxos de trabalho específicos do Astra Trident sejam ativados para registo de rastreio ou supressão de registos.	""
`tridentLogLayers`	Permite que camadas específicas do Astra Trident sejam ativadas para registo de rastreio ou supressão de registos.	""

Opção	Descrição	Padrão
`tridentImage`	Permite a substituição completa da imagem para Astra Trident.	""
`tridentImageTag`	Permite substituir a tag da imagem para Astra Trident.	""
`tridentProbePort`	Permite substituir a porta padrão usada para sondas de disponibilidade/prontidão do Kubernetes.	""
`windows`	Permite que o Astra Trident seja instalado no nó de trabalho do Windows.	`false`
`enableForceDetach`	Permite ativar a função forçar desanexar.	`false`
`excludePodSecurityPolicy`	Exclui a criação da diretiva de segurança do pod do operador.	`false`

Compreensão dos pods dos nós e dos pods do controlador

O Astra Trident é executado como um único pod de controlador, além de um pod de nó em cada nó de trabalho no cluster. O pod de nó deve estar em execução em qualquer host onde você queira potencialmente montar um volume Astra Trident.

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

- O plugin controlador lida com o provisionamento e gerenciamento de volume, como snapshots e redimensionamento.
- O plug-in do nó manipula a conexão do armazenamento ao nó.

O que vem a seguir

Agora você pode "crie um back-end e uma classe de storage, provisione um volume e monte o volume em um pod".

Implantar operador Trident usando Helm (modo off-line)

Você pode implantar o operador Trident e instalar o Astra Trident usando o Helm. Este processo aplica-se a instalações onde as imagens de contentor exigidas pelo Astra Trident são armazenadas em um Registro privado. Se não tiver um registo de imagens privado, utilize o "processo para implantação padrão".

Informações críticas sobre o Astra Trident 23,04

Você deve ler as seguintes informações críticas sobre o Astra Trident.

** informações essenciais sobre o Astra Trident **

- O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Trident antes de atualizar o Kubernetes.
- O Astra Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

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

Usando o Trident"Carta do leme", você pode implantar o operador Trident e instalar o Trident em uma etapa.

Revise "a 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 implantação" que você precisa"Helm versão 3".

Passos

1. Adicione o repositório Astra Trident Helm:

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

2. Use `helm install` e especifique um nome para a localização do Registro de imagens e implantação. O "Imagens de Trident e CSI" pode estar localizado em um Registro ou Registros diferentes, mas todas as imagens CSI devem estar localizadas no mesmo Registro. Nos exemplos `23.04.0`, é a versão do Astra Trident que você está instalando.

Imagens em um Registro

```
helm install <name> netapp-trident/trident-operator --version
23.04.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace>
```

Imagens em diferentes registos

Você deve anexar `sig-storage` ao `imageRegistry` para usar diferentes locais de Registro.

```
helm install <name> netapp-trident/trident-operator --version
23.04.0 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.04.0 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.04 --set tridentImage=<your-
registry>/netapp/trident:23.04.0 --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 de instalação, como nome, namespace, gráfico, status, versão do aplicativo e número de revisão.

Passe os dados de configuração durante a instalação

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

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

Por exemplo, para alterar o valor padrão `debug` do , execute o seguinte `--set` comando onde `23.04.0` está a versão do Astra Trident que você está instalando:

```
helm install <name> netapp-trident/trident-operator --version 23.04.0
--create-namespace --namespace --set tridentDebug=true
```

Opções de configuração

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

Opção	Descrição	Padrão
`nodeSelector`	Etiquetas de nó para atribuição de pod	
`podAnnotations`	Anotações do pod	
`deploymentAnnotations`	Anotações de implantação	
`tolerations`	Tolerâncias para atribuição de pod	
`affinity`	Afinidade para atribuição de pod	
`tridentControllerPluginNodeSelector`	Seletores de nós adicionais para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`tridentControllerPluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`tridentNodePluginNodeSelector`	Seletores de nós adicionais para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`tridentNodePluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`imageRegistry`	Identifica o registo para as `trident-operator`, `trident`e outras imagens. Deixe vazio para aceitar o padrão.	""
`imagePullPolicy`	Define a política de recebimento de imagens para o `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Define os segredos de extração da imagem para as `trident-operator`, `trident` e outras imagens.	
`kubeletDir`	Permite substituir a localização do host do estado interno do kubelet.	""/var/lib/kubelet"

Opção	Descrição	Padrão
`operatorLogLevel`	Permite que o nível de log do operador Trident seja definido como: `trace`, , `debug`, `info` `warn` , , `error` Ou `fatal`.	`"info"`
`operatorDebug`	Permite que o nível de log do operador Trident seja definido como debug.	`true`
`operatorImage`	Permite a substituição completa da imagem para `trident-operator`.	""
`operatorImageTag`	Permite substituir a etiqueta da `trident-operator` imagem.	""
`tridentIPv6`	Permite que o Astra Trident funcione em IPv6 clusters.	`false`
`tridentK8sTimeout`	Substitui o tempo limite padrão de 30 segundos para a maioria das operações da API do Kubernetes (se não for zero, em segundos).	`0`
`tridentHttpRequestTimeout`	Substitui o tempo limite padrão de 90 segundos para as solicitações HTTP, `0s` sendo uma duração infinita para o tempo limite. Valores negativos não são permitidos.	`"90s"`
`tridentSilenceAutosupport`	Permite desativar os relatórios periódicos do Astra Trident AutoSupport.	`false`
`tridentAutosupportImageTag`	Permite substituir a tag da imagem para o contentor Astra Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Permite que o Astra Trident AutoSupport Container ligue para casa por meio de um proxy HTTP.	""
`tridentLogFormat`	Define o formato de registo Astra Trident (`text`ou `json`).	`"text"`
`tridentDisableAuditLog`	Desativa o registrador de auditoria Astra Trident.	`true`
`tridentLogLevel`	Permite que o nível de log do Astra Trident seja definido como: `trace`, `debug`, , `info` `warn` , , `error` `fatal` Ou .	`"info"`
`tridentDebug`	Permite que o nível de log do Astra Trident seja definido como `debug`.	`false`

Opção	Descrição	Padrão
`tridentLogWorkflows`	Permite que fluxos de trabalho específicos do Astra Trident sejam ativados para registo de rastreio ou supressão de registos.	""
`tridentLogLayers`	Permite que camadas específicas do Astra Trident sejam ativadas para registo de rastreio ou supressão de registos.	""
`tridentImage`	Permite a substituição completa da imagem para Astra Trident.	""
`tridentImageTag`	Permite substituir a tag da imagem para Astra Trident.	""
`tridentProbePort`	Permite substituir a porta padrão usada para sondas de disponibilidade/prontidão do Kubernetes.	""
`windows`	Permite que o Astra Trident seja instalado no nó de trabalho do Windows.	`false`
`enableForceDetach`	Permite ativar a função forçar desanexar.	`false`
`excludePodSecurityPolicy`	Exclui a criação da diretiva de segurança do pod do operador.	`false`

Compreensão dos pods dos nós e dos pods do controlador

O Astra Trident é executado como um único pod de controlador, além de um pod de nó em cada nó de trabalho no cluster. O pod de nó deve estar em execução em qualquer host onde você queira potencialmente montar um volume Astra Trident.

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

- O plugin controlador lida com o provisionamento e gerenciamento de volume, como snapshots e redimensionamento.
- O plug-in do nó manipula a conexão do armazenamento ao nó.

O que vem a seguir

Agora você pode "crie um back-end e uma classe de storage, provisione um volume e monte o volume em um pod".

Personalizar a instalação do operador Trident

O operador Trident permite personalizar a instalação do Astra Trident usando os atributos da `TridentOrchestrator` especificação. Se você quiser personalizar a

instalação além do que `TridentOrchestrator` os argumentos permitem, considere usar `tridentctl` para gerar manifestos YAML personalizados para modificar conforme necessário.

Compreensão dos pods dos nós e dos pods do controlador

O Astra Trident é executado como um único pod de controlador, além de um pod de nó em cada nó de trabalho no cluster. O pod de nó deve estar em execução em qualquer host onde você queira potencialmente montar um volume Astra Trident.

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

- O plugin controlador lida com o provisionamento e gerenciamento de volume, como snapshots e redimensionamento.
- O plug-in do nó manipula a conexão do armazenamento ao nó.

Opções de configuração

> `spec.namespace` É especificado em `TridentOrchestrator` para indicar o namespace onde o Astra Trident está instalado. Este parâmetro **não pode ser atualizado após a instalação do Astra Trident**. Tentar fazê-lo faz com que o `TridentOrchestrator` status mude para `Failed`. O Astra Trident não se destina a ser migrado entre namespaces.

Esta tabela detalha `TridentOrchestrator` atributos.

Parâmetro	Descrição	Padrão
`namespace`	Namespace para instalar Astra Trident em	"predefinição"
`debug`	Habilite a depuração para o Astra Trident	falso
`enableForceDetach`	`ontap-san` e `ontap-san-economy` apenas. Funciona com o Kubernetes Non-Graceful Node Shutdown (NGNS) para conceder aos administradores de cluster a capacidade de migrar com segurança cargas de trabalho com volumes montados para novos nós caso um nó não seja saudável. **Esta é uma caraterística experimental em 23,04.** Detalhes sobre Force DetachConsulte para obter detalhes importantes.	`false`
`windows`	A configuração para `true` permite a instalação em nós de trabalho do Windows.	falso
`useIPv6`	Instalar o Astra Trident em IPv6	falso

Parâmetro	Descrição	Padrão
`k8sTimeout`	Tempo limite para operações do Kubernetes	30sec
`silenceAutosupport`	Não envie pacotes AutoSupport para o NetApp automaticamente	falso
`autosupportImage`	A imagem do recipiente para a telemetria AutoSupport	"NetApp/Trident-AutoSupport:23,07"
`autosupportProxy`	O endereço/porta de um proxy para o envio de telemetria AutoSupport	"http://proxy.example.com:8888"
`uninstall`	Um sinalizador usado para desinstalar o Astra Trident	falso
`logFormat`	Formato de log Astra Trident a ser usado [text,json]	"texto"
`tridentImage`	Imagem Astra Trident a instalar	"NetApp/Trident:23,07"
`imageRegistry`	Caminho para o Registro interno, do formato `<registry FQDN>[:port][/subpath]`	"k8s.gcr.io/sig-storage" (mais de k8s 1,19 gb) ou "quay.io/k8scsi gb"
`kubeletDir`	Caminho para o diretório kubelet no host	"/var/lib/kubelet"
`wipeout`	Uma lista de recursos a serem excluídos para realizar uma remoção completa do Astra Trident	
`imagePullSecrets`	Segredos para extrair imagens de um Registro interno	
`imagePullPolicy`	Define a política de recebimento de imagens para o operador Trident. Os valores válidos são: `Always` Para sempre puxar a imagem. `IfNotPresent` para puxar a imagem apenas 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 padrão; opcional
`controllerPluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Segue o mesmo formato que `pod.spec.Tolerations`.	Sem padrão; opcional

Parâmetro	Descrição	Padrão
`nodePluginNodeSelector`	Seletores de nós adicionais para pods. Segue o mesmo formato que `pod.spec.nodeSelector`.	Sem padrão; opcional
`nodePluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Segue o mesmo formato que `pod.spec.Tolerations`.	Sem padrão; opcional

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

Detalhes sobre Force Detach

O Force Detach está disponível apenas para `ontap-san` e. `ontap-san-economy` Antes de ativar a desconexão forçada, o desligamento do nó (NGNS) não gracioso deve ser ativado no cluster do Kubernetes. Para obter mais informações, "Kubernetes: Desligamento do nó não gracioso"consulte .

Como o Astra Trident conta COM NGNS do Kubernetes, não `out-of-service` remova as taints de um nó que não seja saudável até que todos os workloads não toleráveis sejam reprogramados. Aplicar ou remover a taint de forma imprudente pode comprometer a proteção de dados no back-end.

Quando o administrador do cluster do Kubernetes tiver aplicado a `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` alteração ao nó e `enableForceDetach` estiver definido como `true`, o Astra Trident determinará o status do nó e:

1. Cessar o acesso de e/S de back-end para volumes montados nesse nó.
2. Marque o objeto nó Astra Trident como `dirty` (não é seguro para novas publicações).

O controlador Trident rejeitará novas solicitações de volume de publicação até que o nó seja requalificado (depois de ter sido marcado como `dirty`) pelo pod de nó do Trident. Quaisquer cargas de trabalho agendadas com um PVC montado (mesmo depois que o nó do cluster estiver pronto e saudável) não serão aceitas até que o Astra Trident possa verificar o nó `clean` (seguro para novas publicações).

Quando a integridade do nó é restaurada e a taint é removida, o Astra Trident:

1. Identifique e limpe caminhos publicados obsoletos no nó.
2. Se o nó estiver em um `cleanable` estado (a alteração fora de serviço foi removida e o nó está `Ready` no estado) e todos os caminhos obsoletos e publicados estiverem limpos, o Astra Trident reajustará o nó como `clean` e permitirá novos volumes publicados no nó.

Exemplos de configurações

Você pode usar os atributos mencionados acima ao definir `TridentOrchestrator` para personalizar sua instalação.

Exemplo 1: Configuração personalizada básica

Este é um exemplo para uma configuração personalizada básica.

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

Exemplo 2: Implante com seletores de nós

Este exemplo ilustra como o Trident pode ser implantado com seletores de nós:

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

Exemplo 3: Implantar em nós de trabalho do Windows

Este exemplo ilustra a implantação em um nó de trabalho do Windows.

```
cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  windows: true
```

Instale usando o tridentctl

Instale usando o tridentctl

Você pode instalar o Astra Trident usando `tridentctl`o . Este processo aplica-se a instalações onde as imagens de contentor exigidas pelo Astra Trident são armazenadas em um Registro privado ou não. Para personalizar sua `tridentctl` implantação, "Personalizar a implantação do tridentctl"consulte .

Informações críticas sobre o Astra Trident 23,04

Você deve ler as seguintes informações críticas sobre o Astra Trident.

** informações essenciais sobre o Astra Trident **

- O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Trident antes de atualizar o Kubernetes.
- O Astra Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Instalar o Astra Trident usando `tridentctl`

Revise "a 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 trabalho e

"Cluster compatível com Kubernetes"se você tem o Privileges necessário.

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

1. Verifique sua versão do Kubernetes:

```
kubectl version
```

2. Verifique o Privileges do administrador do cluster:

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

3. Verifique se você pode iniciar um pod que usa uma imagem do Docker Hub e alcançar seu sistema de armazenamento pela rede de pod:

```
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 Astra Trident cria um pod 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 de cluster. Baixe e extraia a versão mais recente do instalador do Trident a partir "A seção *assets* no GitHub"do . Atualize _ cliente Trident-installer-XX.XX.X.tar.gz>_ no exemplo com a versão selecionada do Astra Trident.

```
wget https://github.com/NetApp/trident/releases/download/v23.04.0/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
```

Etapa 2: Instale o Astra Trident

Instale o Astra Trident no namespace desejado executando o `tridentctl install` comando. Você pode adicionar argumentos adicionais para especificar a localização do Registro da imagem.

Modo padrão

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

Imagens em um Registro

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:23.04 --trident
-image <your-registry>/trident:23.04.0
```

Imagens em diferentes registos

Você deve anexar `sig-storage` ao `imageRegistry` para usar diferentes locais de Registro.

```
./tridentctl install -n trident --image-registry <your-registry>/sig-
storage --autosupport-image <your-registry>/netapp/trident-
autosupport:23.04 --trident-image <your-
registry>/netapp/trident:23.04.0
```

Seu status de instalação deve ser parecido com isso.

```
....
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=23.04.0
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 Astra Trident foi concluída analisando 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 a 23,01) não tiver um status **Running**, a plataforma não foi instalada. Use `-d` para "ative o modo de depuração" e solucione o problema.

Utilização `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.04.0        | 23.04.0        |
+----------------+----------------+
```

Exemplos de configurações

Exemplo 1: Habilite o Astra Trident a ser executado em nós do Windows

Para permitir que o Astra Trident seja executado em nós do Windows:

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

Exemplo 2: Ativar força desanexar

Para obter mais informações sobre a força desapegada, "Personalizar a instalação do operador Trident"consulte .

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

O que vem a seguir

Agora você pode "crie um back-end e uma classe de storage, provisione um volume e monte o volume em um pod".

Personalize a instalação do tridentctl

Você pode usar o instalador do Astra Trident para personalizar a instalação.

Saiba mais sobre o instalador

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

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

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

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

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

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

O que vem a seguir?

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

Passo 1: Crie um backend

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

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

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

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

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

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

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

Passo 2: Crie uma classe de armazenamento

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

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

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

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

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

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

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

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

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

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

Etapa 3: Provisione seu primeiro volume

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

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

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

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

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

Etapa 4: Montar os volumes em um pod

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

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

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

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

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

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

Para eliminar o volume, elimine a reclamação:

```
kubectl delete pvc basic
```

Agora você pode fazer tarefas adicionais, como as seguintes:

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

Informações sobre direitos autorais

Informações sobre marcas comerciais