Comece agora

Astra Trident

NetApp
February 05, 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 . O Test Drive oferece um ambiente sandbox que inclui um cluster de Kubernetes de três nós e o Astra Trident instalado e configurado. É uma ótima maneira de se familiarizar com o Astra Trident e explorar seus recursos.

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

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

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

Requisitos

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

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

Frontens suportados (orquestradores)

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

- Anthos On-Prem (VMware) e Anthos em bare metal 1,8, 1,9, 1,10
- Kubernetes 1,17 ou posterior (mais recente: 1,23)
- Mecanismo do Kubernetes do Mirantis 3,4
- OpenShift 4,7, 4,8, 4,9

O operador Trident é suportado com estas versões:

- Anthos On-Prem (VMware) e Anthos em bare metal 1,8, 1,9, 1,10
- Kubernetes 1,17 ou posterior (mais recente: 1,23)
- OpenShift 4,7, 4,8, 4,9

> Os usuários do Red Hat OpenShift Container Platform podem observar seu arquivo iniciatorname.iscsi para ficar em branco se estiver usando qualquer versão abaixo de 4,6.8. Este é um bug que foi identificado pela RedHat para ser corrigido com OpenShift 4,6.8. Consulte este "anúncio de correção de bugs". A NetApp recomenda que você use o Astra Trident no OpenShift 4.6.8 e posterior.

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

Backends suportados (armazenamento)

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

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

Requisitos de recursos

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

Recurso	Versão do Kubernetes	É necessário ter portões?
CSI Trident	1,17 e mais tarde	Não
Instantâneos de volume	1,17 e mais tarde	Não
PVC a partir de instantâneos de volume	1,17 e mais tarde	Não
Redimensionamento iSCSI PV	1,17 e mais tarde	Não
ONTAP bidirectional CHAP	1,17 e mais tarde	Não
Políticas de exportação dinâmica	1,17 e mais tarde	Não
Operador Trident	1,17 e mais tarde	Não
Preparação do nó de trabalho automático (beta)	1,17 e mais tarde	Não
Topologia de CSI	1,17 e mais tarde	Não

Sistemas operacionais de host testados

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

Embora o Astra Trident não "ofereça suporte" oficialmente a sistemas operacionais específicos, as seguintes

distribuições Linux são conhecidas por funcionar:

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

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

Configuração de host

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

Configuração do sistema de storage

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

Imagens de contêineres e versões correspondentes do Kubernetes

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

Versão do Kubernetes	Imagem do recipiente
v1.17.0	• NetApp/Trident:22.01.1 • NetApp/Trident-AutoSupport:22,01 • k8s.gcr.io/sig-storage/csi-provisionador:v2,2.2 • k8s.gcr.io/sig-storage/csi-attacher:v3,4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3,0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.4.0 • NetApp/Trident-operador: 22.01.1 (opcional)
v1.18.0	• NetApp/Trident:22.01.1 • NetApp/Trident-AutoSupport:22,01 • k8s.gcr.io/sig-storage/csi-provisionador:v2,2.2 • k8s.gcr.io/sig-storage/csi-attacher:v3,4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3,0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.4.0 • NetApp/Trident-operador: 22.01.1 (opcional)

Versão do Kubernetes	Imagem do recipiente
v1.19.0	• NetApp/Trident:22.01.1 • NetApp/Trident-AutoSupport:22,01 • k8s.gcr.io/sig-storage/csi-provisionador:v2,2.2 • k8s.gcr.io/sig-storage/csi-attacher:v3,4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3,0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.4.0 • NetApp/Trident-operador: 22.01.1 (opcional)
v1.20.0	• NetApp/Trident:22.01.1 • NetApp/Trident-AutoSupport:22,01 • k8s.gcr.io/sig-storage/csi-provisionador:v3,1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3,4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3,0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.4.0 • NetApp/Trident-operador: 22.01.1 (opcional)
v1.21.0	• NetApp/Trident:22.01.1 • NetApp/Trident-AutoSupport:22,01 • k8s.gcr.io/sig-storage/csi-provisionador:v3,1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3,4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3,0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.4.0 • NetApp/Trident-operador: 22.01.1 (opcional)

Versão do Kubernetes	Imagem do recipiente
v1.22.0	• NetApp/Trident:22.01.1 • NetApp/Trident-AutoSupport:22,01 • k8s.gcr.io/sig-storage/csi-provisionador:v3,1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3,4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3,0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.4.0 • NetApp/Trident-operador: 22.01.1 (opcional)
v1.23.0	• NetApp/Trident:22.01.1 • NetApp/Trident-AutoSupport:22,01 • k8s.gcr.io/sig-storage/csi-provisionador:v3,1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3,4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3,0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.4.0 • NetApp/Trident-operador: 22.01.1 (opcional)

No Kubernetes versão 1,20 e posterior, use a imagem validada `k8s.gcr.io/sig-storage/csi-snapshotter:v4.x` somente se a `v1` versão estiver atendendo ao `volumesnapshots.snapshot.storage.k8s.io` CRD. Se a `v1beta1` versão estiver servindo o CRD com/sem a `v1` versão, use a imagem validada `k8s.gcr.io/sig-storage/csi-snapshotter:v3.x`.

Visão geral da implantação

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

Escolha o método de implantação

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

Por que devo usar o operador Trident?

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

O operador Trident fornece vários benefícios, conforme descrito abaixo.

Funcionalidade de autorrecuperação

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

Atualizações fáceis para instalações existentes

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

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

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

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

Manipula automaticamente as atualizações do Kubernetes

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

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

Por que devo usar o Helm?

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

Quando devo usar `tridenctl`?

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

Considerações para mover entre métodos de implantação

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

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

Entenda os modos de implantação

Há três maneiras de implantar o Astra Trident.

Implantação padrão

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

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

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

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

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

Implantação off-line

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

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

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

- "Personalize sua implantação baseada em operador"
*

Se você estiver usando um repositório de imagens privado, adicione `/k8scsi` versões do Kubernetes anteriores a 1,17 ou `/sig-storage` versões do Kubernetes posteriores a 1,17 até o final do URL do Registro privado. Ao usar um Registro privado para `tridentctl` implantação, você deve usar `--trident-image` e `--autosupport-image` em conjunto `--image-registry`com o . Se você estiver implantando o Astra Trident usando o operador Trident, verifique se o orquestrador CR inclui `tridentImage` e `autosupportImage` nos parâmetros de instalação.

Implantação remota

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

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

Outras opções de configuração conhecidas

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

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

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

Implante com o operador Trident

Você pode implantar o Astra Trident com o operador Trident. Você pode implantar o operador Trident manualmente ou usando o Helm.

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

O que você vai precisar

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

- Você tem o Privileges completo para um cluster Kubernetes compatível com Kubernetes que executa o Kubernetes 1,17 e superior.
- Você tem acesso a um sistema de storage NetApp compatível.
- Você tem a capacidade de montar volumes de todos os nós de trabalho do Kubernetes.
- Você tem um host Linux com `kubectl` (ou `oc`, se estiver usando o OpenShift) instalado e configurado para gerenciar o cluster do Kubernetes que deseja usar.

- Você definiu a `KUBECONFIG` variável de ambiente para apontar para a configuração do cluster do Kubernetes.
- Você ativou o "Portas de recurso exigidas pelo Astra Trident".
- Se você estiver usando o Kubernetes com Docker Enterprise "Siga os passos para ativar o acesso CLI", .

Tem tudo isso? Ótimo! Vamos começar.

Implante o operador Trident usando Helm

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

O que você vai precisar

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

- Kubernetes 1,17 e posterior
- Helm versão 3

Passos

1. Adicionar o repositório Helm do Trident:

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

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

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

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

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

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

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

```
$ helm install <name> netapp-trident/trident-operator --version 22.1.0
--set tridentDebug=true
```

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

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

Implante o operador Trident manualmente

Execute as etapas listadas para implantar manualmente o operador Trident.

Etapa 1: Qualifique seu cluster Kubernetes

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

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

Para ver se sua versão do Kubernetes é posterior a 1,17, execute o seguinte comando:

```
kubectl version
```

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

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

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

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

Passo 2: Baixe e configure o operador

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

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

```
wget https://github.com/NetApp/trident/releases/download/v21.04/trident-
installer-21.04.tar.gz
tar -xf trident-installer-21.04.tar.gz
cd trident-installer
```

2. Use o manifesto CRD apropriado para criar o `TridentOrchestrator` CRD. Em seguida, crie um

`TridentOrchestrator` recurso personalizado mais tarde para instanciar uma instalação pelo operador.

Execute o seguinte comando:

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

3. Após a criação do `TridentOrchestrator` CRD, crie os seguintes recursos necessários para a implantação do operador:
   - Um ServiceAccount para o operador
   - Um ClusterRole e ClusterRoleBinding para o ServiceAccount
   - Uma PodSecurityPolicy dedicada
   - O próprio operador

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

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

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

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

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

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

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

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

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

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

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

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

Passo 3: Crie `TridentOrchestrator` e instale o Trident

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

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

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

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

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

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

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

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

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

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

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

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

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

Personalizar a implantação do operador Trident

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

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

Parâmetro	Descrição	Padrão
`namespace`	Namespace para instalar Astra Trident em	"predefinição"
`debug`	Habilite a depuração para o Astra Trident	falso
`IPv6`	Instalar o Astra Trident em IPv6	falso
`k8sTimeout`	Tempo limite para operações do Kubernetes	30sec
`silenceAutosupport`	Não envie pacotes AutoSupport para o NetApp automaticamente	falso
`enableNodePrep`	Gerenciar dependências de nó de trabalho automaticamente (**BETA**)	falso
`autosupportImage`	A imagem do recipiente para a telemetria AutoSupport	"NetApp/Trident-AutoSupport:21.04.0"
`autosupportProxy`	O endereço/porta de um proxy para o envio de telemetria AutoSupport	"http://proxy.example.com:8888"
`uninstall`	Um sinalizador usado para desinstalar o Astra Trident	falso
`logFormat`	Formato de log Astra Trident a ser usado [text,json]	"texto"
`tridentImage`	Imagem Astra Trident a instalar	"NetApp/Trident:21,04"
`imageRegistry`	Caminho para o Registro interno, do formato `<registry FQDN>[:port][/subpath]`	"k8s.gcr.io/sig-storage (mais de k8s 1,17 gb) ou quay.io/k8scsi gb"
`kubeletDir`	Caminho para o diretório kubelet no host	"/var/lib/kubelet"
`wipeout`	Uma lista de recursos a serem excluídos para realizar uma remoção completa do Astra Trident	
`imagePullSecrets`	Segredos para extrair imagens de um Registro interno	

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

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

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

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

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

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

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

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

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

Implante com o tridentctl

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

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

Para personalizar sua implantação, "aqui"consulte .

O que você vai precisar

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

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

Tem tudo isso? Ótimo! Vamos começar.

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

Etapa 1: Qualifique seu cluster Kubernetes

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

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

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

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

```
kubectl version
```

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

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

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

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

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

Passo 2: Baixe e extraia o instalador

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

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

Por exemplo, se a versão mais recente for 21.07.1:

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

Etapa 3: Instale o Astra Trident

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

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

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

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

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

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

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

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

Personalizar a implantação do tridentctl

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

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

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

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

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

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

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

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

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

O que vem a seguir?

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

Passo 1: Crie um backend

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

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

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

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

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

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

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

Passo 2: Crie uma classe de armazenamento

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

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

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

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

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

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

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

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

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

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

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

Etapa 3: Provisione seu primeiro volume

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

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

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

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

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

Etapa 4: Montar os volumes em um pod

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

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

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

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

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

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

Para eliminar o volume, elimine a reclamação:

```
kubectl delete pvc basic
```

Agora você pode fazer tarefas adicionais, como as seguintes:

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

Informações sobre direitos autorais

Informações sobre marcas comerciais