使用 **Trident** 操作员进行部署

Astra Trident

NetApp
April 16, 2024

This PDF was generated from https://docs.netapp.com/zh-cn/trident-2204/trident-get-started/kubernetes-customize-deploy.html on April 16, 2024. Always check docs.netapp.com for the latest.

目录

使用 Trident 操作员进行部署 1
部署Trident操作员并使用Helm安装Astra Trident 1
手动部署 Trident 操作员 2
自定义 Trident 操作员部署 6

使用 Trident 操作员进行部署

您可以使用Trident操作员部署Astra Trident。您可以通过以下两种方式之一部署Trident操作员：

- 使用Trident "Helm图表"：Helm图表将部署Trident操作符并分步安装Trident。
- 手动：Trident提供 "bundle.yaml" 可用于安装操作员和创建关联对象的文件。

> 如果您尚未熟悉 "基本概念"，现在是一个实现这一目标的好时机。

您需要的内容

要部署 Astra Trident ，应满足以下前提条件：

- 您对运行Kubernetes 1.18 - 1.24的受支持Kubernetes集群拥有完全权限。
- 您可以访问受支持的 NetApp 存储系统。
- 您可以从所有 Kubernetes 工作节点挂载卷。
- 您安装了一个安装了 `kubectl` （如果使用的是 OpenShift ，则为 `oc` ）的 Linux 主机，并将其配置为管理要使用的 Kubernetes 集群。
- 您已将 `KUBECONFIG` 环境变量设置为指向 Kubernetes 集群配置。
- 您已启用 "Astra Trident 所需的功能门"。
- 如果您将 Kubernetes 与 Docker Enterprise 结合使用， "按照其步骤启用 CLI 访问"。

明白了吗？太棒了！让我们开始吧。

部署Trident操作员并使用Helm安装Astra Trident

执行列出的步骤以使用 Helm 部署 Trident 操作员。

您需要的内容

除了上述前提条件之外，要使用 Helm 部署 Trident 操作员，还需要满足以下条件：

- Kubernetes 1.18 - 1.24
- Helm 版本 3

步骤

1. 添加 Trident 的 Helm 存储库：

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

2. 使用 `helm install` 命令并为您的部署指定一个名称。请参见以下示例：

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

> 如果尚未为 Trident 创建命名空间，则可以将 ` -create-namespace` 参数添加到 `helm install` 命令中。然后， Helm 将自动为您创建命名空间。

在安装期间，可以通过两种方式传递配置数据：

- ` 值` （或 ` f` ）：指定包含覆盖的 YAML 文件。可以多次指定此值，最右侧的文件将优先。
- ` -set` ：在命令行上指定覆盖。

例如，要更改默认值 `debug` ，请运行以下 ` -set` 命令：

```
$ helm install <name> netapp-trident/trident-operator --version 22.4.0
--set tridentDebug=true
```

Helm 图表中的 `values.yaml` 文件提供了键及其默认值的列表。

`Helm list` 显示有关安装的详细信息，例如名称，命名空间，图表，状态， 应用程序版本，修订版号等。

手动部署 Trident 操作员

执行列出的步骤以手动部署 Trident 操作员。

第 1 步：确定 Kubernetes 集群的资格

首先，您需要登录到 Linux 主机并验证它是否正在管理 _b工作 _ ， "支持的 Kubernetes 集群" 您具有所需权限。

> 使用 OpenShift 时，在以下所有示例中使用 `oc` 而不是 `kubectl` ，并首先以 * 系统： admin* 的身份运行 `oc login -u system ： admin` 或 `oc login -u Kube-admin` 进行登录。

要验证Kubernetes版本、请运行以下命令：

```
kubectl version
```

要查看您是否具有 Kubernetes 集群管理员权限，请运行以下命令：

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

要验证是否可以从 Docker Hub 启动使用映像的 POD 并通过 Pod 网络访问存储系统，请运行以下命令：

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

第 2 步：下载并设置操作员

> 从 21.01 开始， Trident 操作符的范围为集群范围。使用 Trident 操作符安装 Trident 需要创建 `TridentOrchestrator` Custom Resource Definition （ CRD ）并定义其他资源。在安装 Astra Trident 之前，您应执行以下步骤来设置操作员。

1. 下载最新版本的 "Trident 安装程序包" 从 _Downloads_部分 中提取该文件。

```
wget
https://github.com/NetApp/trident/releases/download/v22.04.0/trident-
installer-22.04.0.tar.gz
tar -xf trident-installer-22.04.0.tar.gz
cd trident-installer
```

2. 使用适当的 CRD 清单创建 `TridentOrchestrator` CRD 。然后，您可以稍后创建一个 `TridentOrchestrator` Custom Resource ，以通过操作员实例化安装。

   运行以下命令：

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

3. 创建 `TridentOrchestrator` CRD 后，创建操作员部署所需的以下资源：

   - 操作员的 ServiceAccount
   - 对 ServiceAccount 执行 ClusterRole 和 ClusterRoleBinding
   - 专用的 PodSecurityPolicy
   - 运算符本身

   Trident 安装程序包含用于定义这些资源的清单。默认情况下，操作符部署在 `trident` 命名空间中。如果 `trident` 命名空间不存在，请使用以下清单创建一个。

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

4. 要在非默认 `trident` 命名空间中部署运算符，您应更新 `serviceaccount.yaml` ， `clusterrolebinding.yaml` 和 `operator.yaml` 清单并生成您的 `bundle.yaml` 。

   运行以下命令以更新 YAML 清单并使用 `kucstation.yaml` 生成您的 `bundle.yaml` ：

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

运行以下命令以创建资源并部署操作员：

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

5. 要在部署后验证操作员的状态，请执行以下操作：

```
$ 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
```

操作员部署成功创建了一个在集群中的一个工作节点上运行的 POD 。

> 在 Kubernetes 集群中只能有 * 一个操作符实例 * 。请勿创建 Trident 操作员的多个部署。

第3步：创建 `TridentOrchestrator` 并安装Trident

现在，您可以使用操作员安装 Astra Trident 了！这需要创建 `TridentOrchestrator` 。Trident 安装程序附带了用于创建 `TridentOrchestrator` 的示例定义。这将在 `trident` 命名空间中启动安装。

```
$ 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
```

使用 Trident 操作符，您可以使用 `TridentOrchestrator` 规范中的属性自定义 Astra Trident 的安装方式。请参见 "自定义 Trident 部署"。

状态 `TridentOrchestrator` 指示安装是否成功，并显示已安装的 Trident 版本。

Status	Description
安装	操作员正在使用此 `TridentOrchestrator` CR 安装 Astra Trident 。
已安装	Astra Trident 已成功安装。
正在卸载	操作符正在卸载 Astra Trident ，因为 `sPec.uninstall=true` 。
已卸载	Astra Trident 已卸载。
失败	操作员无法安装，修补，更新或卸载 Astra Trident ；操作员将自动尝试从此状态恢复。如果此状态仍然存在，则需要进行故障排除。
正在更新	操作员正在更新现有安装。
error	不使用 `TridentOrchestrator` 。另一个已存在。

在安装期间， `TridentOrchestrator` 的状态会从 `Installing` 更改为 `Installed` 。如果您观察到 `failed` 状态，并且操作员无法自行恢复，则应检查操作员的日志。请参见 "故障排除" 部分。

您可以通过查看已创建的 Pod 来确认 Astra Trident 安装是否已完成：

```
$ 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
```

您也可以使用 `tridentctl` 检查已安装的 Astra Trident 版本。

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

现在，您可以继续创建后端。请参见 "部署后任务"。

有关在部署期间排除问题的信息，请参见 "故障排除" 部分。

自定义 Trident 操作员部署

使用 Trident 操作符，您可以使用 `TridentOrchestrator` 规范中的属性自定义 Astra Trident 的安装方式。

有关属性列表，请参见下表：

参数	Description	Default
命名空间	用于安装 Astra Trident 的命名空间	default
`debug`	为 Astra Trident 启用调试	false
`IPv6`	安装基于 IPv6 的 Astra Trident	false
`k8sTimeout`	Kubernetes 操作超时	30 秒
`sileAutoSupport`	不要自动向 NetApp 发送 AutoSupport 捆绑包	false
`enableNodePrep`	自动管理工作节点依赖关系（* 测试版 *）	false
`autosupport` 映像	AutoSupport 遥测的容器映像	"NetApp/trident autosupport：21.04.0"
`autosupport` 代理	用于发送 AutoSupport 遥测的代理的地址 / 端口	"http://proxy.example.com:8888"
卸载	用于卸载 Astra Trident 的标志	false
`logFormat`	要使用的 Astra Trident 日志记录格式 [text， json]	文本
`TridentImage`	要安装的 Astra Trident 映像	"NetApp/Trident： 21.04"
`imageRegistry`	内部注册表的路径，格式为 ` < 注册表 FQDN>[： 端口 ]/subpath]`	"K8s.gcr.io/SIG-storage (K8s 1.18+) 或quay.io/k8scsi "
`kubeletDir`	主机上的 kubelet 目录的路径	"/var/lib/kubelet"
`wipeout`	要删除以执行 Astra Trident 完全删除的资源列表	
`imagePullSecs`	从内部注册表中提取映像的机密信息	
控制器插件节点选择器	运行 Trident 控制器 CSI 插件的 Pod 的其他节点选择器。格式与 pod.spec.nodeSelector 相同。	无默认值；可选
控制器插件公差	覆盖运行 Trident 控制器 CSI 插件的 Pod 的容错。格式与 po.spec.Tolerations 相同。	无默认值；可选
`nodePluginNodeSelector`	运行 Trident Node CSI 插件的 Pod 的其他节点选择器。格式与 pod.spec.nodeSelector 相同。	无默认值；可选
`nodePluginTholeations`	覆盖运行 Trident Node CSI 插件的 Pod 的容错。格式与 po.spec.Tolerations 相同。	无默认值；可选

在 `TridentOrchestrator` 中指定 `spec.namespace` ，以表示安装了哪个命名空间 Astra Trident 。此参数 * 安装 Astra Trident 后无法更新 * 。尝试执行此操作会导致 `TridentOrchestrator` 的状态更改为 `Failed` 。Astra Trident 不适用于跨命名空间迁移。

自动员工节点准备是一项 * 测试版功能 * ，仅用于非生产环境。

有关格式化 POD 参数的详细信息，请参见 "将 Pod 分配给节点"。

在定义 `TridentOrchestrator` 时，您可以使用上述属性自定义安装。以下是一个示例：

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

下面是另一个示例，说明如何使用节点选择器部署 Trident：

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

如果您希望自定义安装超出 `TridentOrchestrator` 参数的允许范围，则应考虑使用 `tridentctl` 生成自定义 YAML 清单，以便您可以根据需要进行修改。

版权信息

版权所有 © 2024 NetApp, Inc.。保留所有权利。中国印刷。未经版权所有者事先书面许可，本文档中受版权保护的任何部分不得以任何形式或通过任何手段（图片、电子或机械方式，包括影印、录音、录像或存储在电子检索系统中）进行复制。

从受版权保护的 NetApp 资料派生的软件受以下许可和免责声明的约束：

本软件由 NetApp 按“原样”提供，不含任何明示或暗示担保，包括但不限于适销性以及针对特定用途的适用性的隐含担保，特此声明不承担任何责任。在任何情况下，对于因使用本软件而以任何方式造成的任何直接性、间接性、偶然性、特殊性、惩罚性或后果性损失（包括但不限于购买替代商品或服务；使用、数据或利润方面的损失；或者业务中断），无论原因如何以及基于何种责任理论，无论出于合同、严格责任或侵权行为（包括疏忽或其他行为），NetApp 均不承担责任，即使已被告知存在上述损失的可能性。

NetApp 保留在不另行通知的情况下随时对本文档所述的任何产品进行更改的权利。除非 NetApp 以书面形式明确同意，否则 NetApp 不承担因使用本文档所述产品而产生的任何责任或义务。使用或购买本产品不表示获得 NetApp 的任何专利权、商标权或任何其他知识产权许可。

本手册中描述的产品可能受一项或多项美国专利、外国专利或正在申请的专利的保护。

有限权利说明：政府使用、复制或公开本文档受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中“技术数据权利 — 非商用”条款第 (b)(3) 条规定的限制条件的约束。

本文档中所含数据与商业产品和/或商业服务（定义见 FAR 2.101）相关，属于 NetApp, Inc. 的专有信息。根据本协议提供的所有 NetApp 技术数据和计算机软件具有商业性质，并完全由私人出资开发。 美国政府对这些数据的使用权具有非排他性、全球性、受限且不可撤销的许可，该许可既不可转让，也不可再许可，但仅限在与交付数据所依据的美国政府合同有关且受合同支持的情况下使用。除本文档规定的情形外，未经 NetApp, Inc. 事先书面批准，不得使用、披露、复制、修改、操作或显示这些数据。美国政府对国防部的授权仅限于 DFARS 的第 252.227-7015(b)（2014 年 2 月）条款中明确的权利。

商标信息

NetApp、NetApp 标识和 http://www.netapp.com/TM 上所列的商标是 NetApp, Inc. 的商标。其他公司和产品名称可能是其各自所有者的商标。