使用 Trident 操作员进行部署

Astra Trident

NetApp
April 16, 2024

目录

使用 Trident 操作员进行部署

您可以使用 Trident 操作员部署 Astra Trident 。您可以手动或使用 Helm 部署 Trident 操作员。

> 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 1.17 及更高版本的受支持 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。

使用 Helm 部署 Trident 操作员

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 1.17 及更高版本
- Helm 版本 3

步骤

1. 从下载安装程序包 "Trident GitHub" 页面。安装程序包的 ` /helm` 目录中包含 Helm 图表。
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 <name> trident-operator-21.07.1.tgz --namespace <namespace
   you want to use for Trident>
   ```

   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> trident-operator-21.07.1.tgz --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。

要查看您的 Kubernetes 版本是否高于 1.17 ，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/v21.04/trident-
installer-21.04.tar.gz
tar -xf trident-installer-21.04.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

通过 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 1.17+）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	

> 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。

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
  tridentImage: netapp/trident:21.04.0
  imagePullSecrets:
  - thisisasecret
```

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。