



使用Trident操作员安装 Astra Trident

NetApp
June 28, 2024

目录

使用Trident操作员安装	1
手动部署Trident操作员(标准模式)	1
手动部署Trident操作员(脱机模式)	6
使用Helm部署Trident操作员(标准模式)	12
使用Helm部署Trident操作员(脱机模式)	15
自定义Trident操作员安装	19

使用Trident操作员安装

手动部署Trident操作员(标准模式)

您可以手动部署Trident操作员以安装Astra Trident。此过程将处理适用场景 安装、其中、Astra Trident所需的容器映像不会存储在专用注册表中。如果您有专用映像注册表、请使用 ["脱机部署过程"](#)。

有关Astra Trident的重要信息24.02

您必须阅读以下有关Astra Trident的重要信息。

**** 中有关Astra **** 的信息

- 现在、在Trident中支持Kubnetes 1.27。在升级Kubernetes之前升级Trident。
- Astra Trident会严格强制在SAN环境中使用多路径配置、建议值为 `find_multipaths: no` 在 `multipath.conf` 文件中。

使用非多路径配置或 `find_multipaths: yes` 或 `find_multipaths: smart` `multipath.conf` 文件中的值将导致挂载失败。Trident已建议使用 `find_multipaths: no` 自21.07版起。

手动部署Trident操作员并安装Trident

请查看 ["安装概述"](#) 以确保满足安装前提条件并为您的环境选择正确的安装选项。

开始之前

开始安装之前、请登录到Linux主机并验证它是否正在管理一个正常运行的、["支持的 Kubernetes 集群"](#) 并且您拥有必要的特权。



使用OpenShift `oc` 而不是 `kubectl` 在下面的所有示例中、运行以*系统: admin*身份登录 `oc login -u system:admin` 或 `oc login -u kube-admin`。

1. 验证Kubernetes版本:

```
kubectl version
```

2. 验证集群管理员权限:

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

3. 验证您是否可以从Docker Hub启动使用映像的POD并通过POD网络访问存储系统:

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

第1步: 下载Trident安装程序包

Astra Trident安装程序包包含部署Trident操作员和安装Astra Trident所需的所有内容。从下载并提取最新版本的Trident安装程序 "[GitHub上的_assets_部分](#)"。

```
wget https://github.com/NetApp/trident/releases/download/v24.02.0/trident-
installer-24.02.0.tar.gz
tar -xf trident-installer-24.02.0.tar.gz
cd trident-installer
```

第2步: 创建 TridentOrchestrator CRD

创建 TridentOrchestrator 自定义资源定义(CRD)。您将创建 TridentOrchestrator 稍后自定义资源。使用中相应的CRD YAML版本 `deploy/crds` 以创建 TridentOrchestrator CRD。

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

第3步: 部署Trident操作员

Astra Trident安装程序提供了一个包文件、可用于安装操作员和创建关联对象。使用此捆绑包文件可以轻松部署操作员并使用默认配置安装Astra Trident。

- 对于运行Kubernetes 1.24或更早版本的集群、请使用 `bundle_pre_1_25.yaml`。
- 对于运行Kubernetes 1.25或更高版本的集群、请使用 `bundle_post_1_25.yaml`。

开始之前

- 默认情况下、通过使用三端安装程序、可以在中部署操作员 `trident` 命名空间。如果 `trident` 命名空间不存在、请使用以下命令创建命名空间：

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

- 在非命名空间中部署操作员 `trident` 命名空间、更新 `serviceaccount.yaml`、`clusterrolebinding.yaml` 和 `operator.yaml` 并使用生成捆绑包文件 `kustomization.yaml`。
 - a. 创建 `kustomization.yaml` 使用以下命令、其中 `<bundle.yaml>` 为 `bundle_pre_1_25.yaml` 或 `bundle_post_1_25.yaml` 根据您的 Kubernetes 版本。

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

- b. 使用以下命令编译分发包、其中 `_Data <bundle.yaml>` 是 `bundle_pre_1_25.yaml` 或 `bundle_post_1_25.yaml` 根据您的 Kubernetes 版本。

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

步骤

1. 创建资源并部署操作员：

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

2. 验证是否已创建操作员、部署和副本集。

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



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4步：创建 `TridentOrchestrator` 并安装 **Trident**

现在、您可以创建 `TridentOrchestrator` 并安装 `Astra Trident`。您也可以选择 ["自定义Trident安装"](#) 使用中的属性 `TridentOrchestrator` 规格

```

kubect1 create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubect1 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:24.02
    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:24.02.0
  Message:            Trident installed Namespace:
trident
  Status:              Installed
  Version:             v24.02.0
Events:
  Type Reason Age From Message ---- -
-----
Normal
Installing 74s trident-operator.netapp.io Installing Trident Normal
Installed 67s trident-operator.netapp.io Trident installed

```

验证安装。

可以通过多种方法验证您的安装。

使用 TridentOrchestrator status

的状态 TridentOrchestrator 指示安装是否成功、并显示已安装的Trident版本。在安装期间、的状态 TridentOrchestrator 更改自 Installing to Installed。如果您观察到 Failed 状态、并且操作员无法自行恢复、"检查日志"。

Status	Description
安装	操作员正在使用此安装Astra Trident TridentOrchestrator CR.
已安装	Astra Trident 已成功安装。
正在卸载	操作员正在卸载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三端测试；操作员将自动尝试从该状态中恢复。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	。TridentOrchestrator 未使用。已经是另一个存在。

正在使用POD创建状态

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

```
kubectl get pods -n trident
```

NAME	READY	STATUS	RESTARTS
AGE			
trident-controller-7d466bf5c7-v4cpw 1m	6/6	Running	0
trident-node-linux-mr6zc 1m	2/2	Running	0
trident-node-linux-xrp7w 1m	2/2	Running	0
trident-node-linux-zh2jt 1m	2/2	Running	0
trident-operator-766f7b8658-ldzsv 3m	1/1	Running	0

使用 tridentctl

您可以使用 tridentctl 检查安装的Astra Trident版本。

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

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 24.02.0        | 24.02.0        |
+-----+-----+
```

手动部署Trident操作员(脱机模式)

您可以手动部署Trident操作员以安装Astra Trident。此过程将处理适用场景 安装、其中、Astra Trident所需的容器映像存储在专用注册表中。如果您没有专用映像注册表、请使用 ["标准部署流程"](#)。

有关Astra Trident的重要信息24.02

您必须阅读以下有关Astra Trident的重要信息。

**** 中有关Astra **** 的信息

- 现在、在Trident中支持Kubnetes 1.27。在升级Kubernetes之前升级Trident。
- Astra Trident会严格强制在SAN环境中使用多路径配置、建议值为 `find_multipaths: no` 在 `multipath.conf` 文件中。

使用非多路径配置或 `find_multipaths: yes` 或 `find_multipaths: smart` `multipath.conf` 文件中的值将导致挂载失败。Trident已建议使用 `find_multipaths: no` 自21.07版起。

手动部署Trident操作员并安装Trident

请查看 ["安装概述"](#) 以确保满足安装前提条件并为您的环境选择正确的安装选项。

开始之前

登录到Linux主机并验证它是否正在管理正常工作的和 ["支持的 Kubernetes 集群"](#) 并且您拥有必要的特权。



使用OpenShift `oc` 而不是 `kubectl` 在下面的所有示例中、运行以*系统: admin*身份登录 `oc login -u system:admin` 或 `oc login -u kube-admin`。

1. 验证Kubernetes版本:

```
kubectl version
```

2. 验证集群管理员权限:

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

3. 验证您是否可以从Docker Hub启动使用映像的POD并通过POD网络访问存储系统:

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

第1步: 下载Trident安装程序包

Astra Trident安装程序包包含部署Trident操作员和安装Astra Trident所需的所有内容。从下载并提取最新版本的Trident安装程序 "[GitHub上的_assets_部分](#)"。

```
wget https://github.com/NetApp/trident/releases/download/v24.02.0/trident-
installer-24.02.0.tar.gz
tar -xf trident-installer-24.02.0.tar.gz
cd trident-installer
```

第2步: 创建 TridentOrchestrator CRD

创建 TridentOrchestrator 自定义资源定义(CRD)。您将创建 TridentOrchestrator 稍后自定义资源。使用中相应的CRD YAML版本 `deploy/crds` 以创建 TridentOrchestrator CRD:

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

第3步: 更新操作符中的注册表位置

在中 `/deploy/operator.yaml`、更新 `image: docker.io/netapp/trident-operator:24.02.0` 以反映映像注册表的位置。Your "[Trident和CSI映像](#)" 可以位于一个注册表或不同的注册表中、但所有CSI映像都必须位于同一注册表中。例如:

- `image: <your-registry>/trident-operator:24.02.0` 如果您的映像全部位于一个注册表中。
- `image: <your-registry>/netapp/trident-operator:24.02.0` 如果Trident映像与CSI映像位于不同的注册表中。

第4步：部署TRident操作员

Astra Trident安装程序提供了一个包文件、可用于安装操作员和创建关联对象。使用此捆绑包文件可以轻松部署操作员并使用默认配置安装Astra Trident。

- 对于运行Kubernetes 1.24或更早版本的集群、请使用 `bundle_pre_1_25.yaml`。
- 对于运行Kubernetes 1.25或更高版本的集群、请使用 `bundle_post_1_25.yaml`。

开始之前

- 默认情况下、通过使用三端安装程序、可以在中部署操作员 `trident` 命名空间。如果 `trident` 命名空间不存在、请使用以下命令创建命名空间：

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

- 在非命名空间中部署操作员 `trident` 命名空间、更新 `serviceaccount.yaml`、`clusterrolebinding.yaml` 和 `operator.yaml` 并使用生成捆绑包文件 `kustomization.yaml`。
 - a. 创建 `kustomization.yaml` 使用以下命令、其中 `<bundle.yaml>` 为 `bundle_pre_1_25.yaml` 或 `bundle_post_1_25.yaml` 根据您的Kubernetes版本。

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

- b. 使用以下命令编译分发包、其中 `_Data <bundle.yaml>` 是 `bundle_pre_1_25.yaml` 或 `bundle_post_1_25.yaml` 根据您的Kubernetes版本。

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

步骤

1. 创建资源并部署操作员：

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

2. 验证是否已创建操作员、部署和副本集。

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



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5步：在中更新映像注册表位置 TridentOrchestrator

Your "**Trident和CSI映像**" 可以位于一个注册表或不同的注册表中、但所有CSI映像都必须位于同一注册表中。更新 `deploy/crds/tridentorchestrator_cr.yaml` 根据注册表配置添加其他位置规格。

一个注册表中的映像

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:24.02"
tridentImage: "<your-registry>/trident:24.02.0"
```

不同注册表中的映像

您必须附加 sig-storage 到 imageRegistry 使用不同的注册表位置。

```
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:24.02"
tridentImage: "<your-registry>/netapp/trident:24.02.0"
```

第6步：创建 TridentOrchestrator 并安装Trident

现在、您可以创建 TridentOrchestrator 并安装Astra Trident。您也可以选择继续操作 ["自定义Trident安装"](#) 使用中的属性 TridentOrchestrator 规格以下示例显示了Trident和CSI映像位于不同注册表中的安装。

```

kubect1 create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubect1 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:24.02
  Debug:             true
  Image Registry:    <your-registry>/sig-storage
  Namespace:        trident
  Trident Image:     <your-registry>/netapp/trident:24.02.0
Status:
  Current Installation Params:
    IPv6:            false
    Autosupport Hostname:
    Autosupport Image: <your-registry>/netapp/trident-
autosupport:24.02
    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:24.02.0
  Message:           Trident installed
  Namespace:         trident
  Status:            Installed
  Version:           v24.02.0
Events:
  Type Reason Age From Message ---- -
-----
Normal
Installing 74s trident-operator.netapp.io Installing Trident Normal
Installed 67s trident-operator.netapp.io Trident installed

```

验证安装。

可以通过多种方法验证您的安装。

使用 `TridentOrchestrator status`

的状态 `TridentOrchestrator` 指示安装是否成功、并显示已安装的Trident版本。在安装期间、的状态 `TridentOrchestrator` 更改自 `Installing` to `Installed`。如果您观察到 `Failed` 状态、并且操作员无法自行恢复、["检查日志"](#)。

Status	Description
安装	操作员正在使用此安装Astra Trident TridentOrchestrator CR.
已安装	Astra Trident 已成功安装。
正在卸载	操作员正在卸载Astra Trident、因为 <code>spec.uninstall=true</code> 。
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三端测试；操作员将自动尝试从该状态中恢复。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	。 <code>TridentOrchestrator</code> 未使用。已经是另一个存在。

正在使用POD创建状态

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

```
kubectl get pods -n trident
```

NAME	READY	STATUS	RESTARTS
AGE			
trident-controller-7d466bf5c7-v4cpw 1m	6/6	Running	0
trident-node-linux-mr6zc 1m	2/2	Running	0
trident-node-linux-xrp7w 1m	2/2	Running	0
trident-node-linux-zh2jt 1m	2/2	Running	0
trident-operator-766f7b8658-ldzsv 3m	1/1	Running	0

使用 `tridentctl`

您可以使用 `tridentctl` 检查安装的Astra Trident版本。

```
./tridentctl -n trident version

+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 24.02.0       | 24.02.0       |
+-----+-----+
```

使用Helm部署Trident操作员(标准模式)

您可以使用Helm部署Trident操作员并安装Astra Trident。此过程将处理适用场景 安装、其中、Astra Trident所需的容器映像不会存储在专用注册表中。如果您有专用映像注册表、请使用 ["脱机部署过程"](#)。

有关Astra Trident的重要信息24.02

您必须阅读以下有关Astra Trident的重要信息。

**** 中有关Astra **** 的信息

- 现在、在Trident中支持Kubernetes 1.27。在升级Kubernetes之前升级Trident。
- Astra Trident会严格强制在SAN环境中使用多路径配置、建议值为 `find_multipaths: no` 在 `multipath.conf` 文件中。

使用非多路径配置或 `find_multipaths: yes` 或 `find_multipaths: smart` `multipath.conf` 文件中的值将导致挂载失败。Trident已建议使用 `find_multipaths: no` 自21.07版起。

部署Trident操作员并使用Helm安装Astra Trident

使用Trident ["Helm图表"](#) 您可以一步部署Trident操作员并安装Trident。

请查看 ["安装概述"](#) 以确保满足安装前提条件并为您的环境选择正确的安装选项。

开始之前

除了 ["部署前提条件"](#) 您需要 ["Helm 版本 3"](#)。

步骤

1. 添加Astra Trident Helm存储库：

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

2. 使用 `helm install` 并为您的部署指定一个名称、如以下示例中所示 100.2402.0 是您要安装的Astra Trident版本。

```
helm install <name> netapp-trident/trident-operator --version 100.2402.0  
--create-namespace --namespace <trident-namespace>
```



如果您已为Trident创建命名空间、则会显示 `--create-namespace` 参数不会创建其他命名空间。

您可以使用 `helm list` 查看安装详细信息、例如名称、命名空间、图表、状态、应用程序版本、和修订版本号。

在安装期间传递配置数据

在安装期间，可以通过两种方式传递配置数据：

选项	Description
<code>--values</code> (或 <code>-f</code>)	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。
<code>--set</code>	在命令行上指定覆盖。

例如、要更改的默认值 `debug`、运行以下命令 `--set` 命令位置 100.2402.0 是您要安装的Astra Trident版本：

```
helm install <name> netapp-trident/trident-operator --version 100.2402.0  
--create-namespace --namespace trident --set tridentDebug=true
```

配置选项

此表和 `values.yaml` 文件(属于Helm图表的一部分)提供了键列表及其默认值。

选项	Description	Default
<code>nodeSelector</code>	用于POD分配的节点标签	
<code>podAnnotations</code>	POD标注	
<code>deploymentAnnotations</code>	部署标注	
<code>tolerations</code>	POD分配的差值	
<code>affinity</code>	用于Pod分配的相关性	

选项	Description	Default
tridentControllerPluginNodeSelector	Pod的其他节点选择器。请参见 了解控制器Pod和节点Pod 了解详细信息。	
tridentControllerPluginTolerations	覆盖Kubernetes对Pod的容错。请参见 了解控制器Pod和节点Pod 了解详细信息。	
tridentNodePluginNodeSelector	Pod的其他节点选择器。请参见 了解控制器Pod和节点Pod 了解详细信息。	
tridentNodePluginTolerations	覆盖Kubernetes对Pod的容错。请参见 了解控制器Pod和节点Pod 了解详细信息。	
imageRegistry	标识的注册表 trident-operator, `trident` 和其他图像。留空以接受默认值。	""
imagePullPolicy	设置的映像提取策略 trident-operator。	IfNotPresent
imagePullSecrets	设置的映像提取密钥 trident-operator, `trident` 和其他图像。	
kubeletDir	允许覆盖kubelet内部状态的主机位置。	"/var/lib/kubelet"
operatorLogLevel	允许将Trident操作符的日志级别设置为: trace, debug, info, warn, error`或`fatal。	"info"
operatorDebug	允许将Trident操作符的日志级别设置为DEBUG。	true
operatorImage	允许完全覆盖的映像 trident-operator。	""
operatorImageTag	允许覆盖的标记 trident-operator 图像。	""
tridentIPv6	允许在IPv6集群中使用Astra Trident。	false
tridentK8sTimeout	覆盖大多数Kubernetes API操作的默认30秒超时(如果不为零、则以秒为单位)。	0
tridentHttpRequestTimeout	使用覆盖HTTP请求的默认90秒超时 0s 为超时的无限持续时间。不允许使用负值。	"90s"
tridentSilenceAutoSupport	允许禁用Astra Trident定期AutoSupport 报告。	false
tridentAutoSupportImageTag	允许覆盖Astra Trident AutoSupport 容器的映像标记。	<version>
tridentAutoSupportProxy	允许Astra Trident AutoSupport 容器通过HTTP代理进行回拨。	""
tridentLogFormat	设置Astra Trident日志记录格式 (text 或 json) 。	"text"
tridentDisableAuditLog	禁用Astra Trident审核日志程序。	true
tridentLogLevel	允许将Astra Trident的日志级别设置为: trace, debug, info, warn, error`或`fatal。	"info"
tridentDebug	允许将Astra Trident的日志级别设置为 debug。	false
tridentLogWorkflows	允许为跟踪日志记录或日志禁止启用特定的Astra Trident工作流。	""

选项	Description	Default
tridentLogLayers	允许为跟踪日志记录或日志禁止启用特定的Astra Trident层。	""
tridentImage	允许完全覆盖Astra Trident的映像。	""
tridentImageTag	允许覆盖Astra Trident的映像标记。	""
tridentProbePort	允许覆盖用于Kubernetes活动/就绪性探测的默认端口。	""
windows	允许在Windows工作节点上安装Astra Trident。	false
enableForceDetach	允许启用强制分离功能。	false
excludePodSecurityPolicy	从创建过程中排除操作员POD安全策略。	false
cloudProvider	设置为 "Azure" 在AKS集群上使用托管身份或云身份时。如果在EKS集群上使用云标识、请设置为"Aws"。	""
cloudIdentity	在AKS集群上使用云标识时、设置为工作负载标识("azure.Workload .Identity /client-id: XXXXXX-xxxx-xxxx-xxxx-xxxxxxxxxxxxx")。在EKS集群上使用云身份时、设置为AWS IAM角色("eks.amazonaws.com/role-arn: arn: aws: iam: : : 12356: roe/astratriident-roe")。	""
iscsiSelfHealingInterval	调用iSCSI自我修复的间隔。	5m0s
iscsiSelfHealingWaitTime	iSCSI自我修复通过执行注销和后续登录来尝试解决陈旧会话之前经过的持续时间。	7m0s

了解控制器Pod和节点Pod

Astra Trident作为一个控制器POD运行、并在集群中的每个工作节点上运行一个节点POD。节点POD必须在任何可能要挂载Astra Trident卷的主机上运行。

Kubernetes ["节点选择器"](#) 和 ["容忍和损害"](#) 用于限制Pod在特定节点或首选节点上运行。使用`ControllerPlugin`和 NodePlugin、您可以指定约束和覆盖。

- 控制器插件负责卷配置和管理、例如快照和调整大小。
- 节点插件负责将存储连接到节点。

使用Helm部署Trident操作员(脱机模式)

您可以使用Helm部署Trident操作员并安装Astra Trident。此过程将处理适用场景 安装、其中、Astra Trident所需的容器映像存储在专用注册表中。如果您没有专用映像注册表、请使用 ["标准部署流程"](#)。

有关Astra Trident的重要信息24.02

您必须阅读以下有关Astra Trident的重要信息。

**** 中有关Astra **** 的信息

- 现在、在Trident中支持Kubnetes 1.27。在升级Kubernetes之前升级Trident。
- Astra Trident会严格强制在SAN环境中使用多路径配置、建议值为 `find_multipaths: no` 在 `multipath.conf` 文件中。

使用非多路径配置或 `find_multipaths: yes` 或 `find_multipaths: smart` `multipath.conf` 文件中的值将导致挂载失败。Trident已建议使用 `find_multipaths: no` 自21.07版起。

部署Trident操作员并使用Helm安装Astra Trident

使用Trident "[Helm图表](#)" 您可以一步部署Trident操作员并安装Trident。

请查看 "[安装概述](#)" 以确保满足安装前提条件并为您的环境选择正确的安装选项。

开始之前

除了 "[部署前提条件](#)" 您需要 "[Helm 版本 3](#)"。

步骤

1. 添加Astra Trident Helm存储库:

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

2. 使用 `...helm install` 并为部署和映像注册表位置指定一个名称。Your "[Trident和CSI映像](#)" 可以位于一个注册表或不同的注册表中、但所有CSI映像都必须位于同一注册表中。在示例中、`100.2402.0` 是您要安装的Astra Trident版本。

一个注册表中的映像

```
helm install <name> netapp-trident/trident-operator --version
100.2402.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace>
```

不同注册表中的映像

您必须附加 sig-storage 到 imageRegistry 使用不同的注册表位置。

```
helm install <name> netapp-trident/trident-operator --version
100.2402.0 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:24.02.0 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:24.02 --set tridentImage=<your-
registry>/netapp/trident:24.02.0 --create-namespace --namespace
<trident-namespace>
```



如果您已为Trident创建命名空间、则会显示 `--create-namespace` 参数不会创建其他命名空间。

您可以使用 `helm list` 查看安装详细信息、例如名称、命名空间、图表、状态、应用程序版本、和修订版本号。

在安装期间传递配置数据

在安装期间，可以通过两种方式传递配置数据：

选项	Description
<code>--values</code> (或 <code>-f</code>)	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。
<code>--set</code>	在命令行上指定覆盖。

例如、要更改的默认值 `debug`、运行以下命令 `--set` 命令位置 `100.2402.0` 是您要安装的Astra Trident版本：

```
helm install <name> netapp-trident/trident-operator --version 100.2402.0
--create-namespace --namespace trident --set tridentDebug=true
```

配置选项

此表和 `values.yaml` 文件(属于Helm图表的一部分)提供了键列表及其默认值。

选项	Description	Default
nodeSelector	用于POD分配的节点标签	
podAnnotations	POD标注	
deploymentAnnotations	部署标注	
tolerations	POD分配的差值	
affinity	用于Pod分配的相关性	
tridentControllerPluginNodeSelector	Pod的其他节点选择器。请参见 "了解控制器Pod和节点Pod" 了解详细信息。	
tridentControllerPluginTolerations	覆盖Kubernetes对Pod的容错。请参见 "了解控制器Pod和节点Pod" 了解详细信息。	
tridentNodePluginNodeSelector	Pod的其他节点选择器。请参见 "了解控制器Pod和节点Pod" 了解详细信息。	
tridentNodePluginTolerations	覆盖Kubernetes对Pod的容错。请参见 "了解控制器Pod和节点Pod" 了解详细信息。	
imageRegistry	标识的注册表 trident-operator, `trident` 和其他图像。留空以接受默认值。	""
imagePullPolicy	设置的映像提取策略 trident-operator。	IfNotPresent
imagePullSecrets	设置的映像提取密钥 trident-operator, `trident` 和其他图像。	
kubeletDir	允许覆盖kubelet内部状态的主机位置。	"/var/lib/kubelet"
operatorLogLevel	允许将Trident操作符的日志级别设置为: trace, debug, info, warn, error`或`fatal。	"info"
operatorDebug	允许将Trident操作符的日志级别设置为DEBUG。	true
operatorImage	允许完全覆盖的映像 trident-operator。	""
operatorImageTag	允许覆盖的标记 trident-operator 图像。	""
tridentIPv6	允许在IPv6集群中使用Astra Trident。	false
tridentK8sTimeout	覆盖大多数Kubernetes API操作的默认30秒超时(如果不为零、则以秒为单位)。	0

选项	Description	Default
tridentHttpRequestTimeout	使用覆盖HTTP请求的默认90秒超时0s 为超时的无限持续时间。不允许使用负值。	"90s"
tridentSilenceAutosupport	允许禁用Astra Trident定期AutoSupport 报告。	false
tridentAutosupportImageTag	允许覆盖Astra Trident AutoSupport 容器的映像标记。	<version>
tridentAutosupportProxy	允许Astra Trident AutoSupport 容器通过HTTP代理进行回拨。	""
tridentLogFormat	设置Astra Trident日志记录格式(text 或 json) 。	"text"
tridentDisableAuditLog	禁用Astra Trident审核日志程序。	true
tridentLogLevel	允许将Astra Trident的日志级别设置为: trace, debug, info, warn, error`或`fatal。	"info"
tridentDebug	允许将Astra Trident的日志级别设置为 debug。	false
tridentLogWorkflows	允许为跟踪日志记录或日志禁止启用特定的Astra Trident工作流。	""
tridentLogLayers	允许为跟踪日志记录或日志禁止启用特定的Astra Trident层。	""
tridentImage	允许完全覆盖Astra Trident的映像。	""
tridentImageTag	允许覆盖Astra Trident的映像标记。	""
tridentProbePort	允许覆盖用于Kubernetes活动/就绪性探测的默认端口。	""
windows	允许在Windows工作节点上安装Astra Trident。	false
enableForceDetach	允许启用强制分离功能。	false
excludePodSecurityPolicy	从创建过程中排除操作员POD安全策略。	false

自定义Trident操作员安装

使用Trident操作员可以使用中的属性自定义Astra Trident安装 TridentOrchestrator 规格如果您要对安装进行自定义、使其超出预期范围 TridentOrchestrator 参数允许、请考虑使用 tridentctl 生成自定义YAML清单以根据需要进行修改。

了解控制器Pod和节点Pod

Astra Trident作为一个控制器POD运行、并在集群中的每个工作节点上运行一个节点POD。节点POD必须在任何可能要挂载Astra Trident卷的主机上运行。

Kubernetes "节点选择器" 和 "容忍和损害" 用于限制Pod在特定节点或首选节点上运行。使用`ControllerPlugin` 和 NodePlugin、您可以指定约束和覆盖。

- 控制器插件负责卷配置和管理、例如快照和调整大小。
- 节点插件负责将存储连接到节点。

配置选项



`spec.namespace` 在中指定 `TridentOrchestrator` 表示安装了Astra Trident的命名空间。此参数 * 安装 Astra Trident 后无法更新 *。如果尝试执行此操作、则会导致 `TridentOrchestrator` 要更改为的状态 `Failed`。Astra Trident不能跨命名空间迁移。

此表详细介绍了相关信息 `TridentOrchestrator` 属性。

参数	Description	Default
<code>namespace</code>	用于安装 Astra Trident 的命名空间	"default"
<code>debug</code>	为 Astra Trident 启用调试	false
<code>enableForceDetach</code>	ontap-san 和 ontap-san-economy 仅限。 可与Kubnetes非正常节点关闭(NGN)结合使用、使集群管理员能够在节点运行状况不正常时将已挂载卷的工作负载安全地迁移到新节点。	false
<code>windows</code>	设置为 true 启用在Windows工作节点上的安装。	false
<code>cloudProvider</code>	设置为 "Azure" 在AKS集群上使用托管身份或云身份时。如果在EKS集群上使用云标识、请设置为"Aws"。	" "
<code>cloudIdentity</code>	在AKS集群上使用云标识时、设置为工作负载标识("azure.Workload .Identity /client-id: XXXXXX-xxxx-xxxx-xxxx-xxxxxxxxxxxxx")。在EKS集群上使用云身份时、设置为AWS IAM角色("eks.amazonaws.com/role-arn: arn: aws: iam: : : 12356: roe/astratriident-roe")。	" "
<code>IPv6</code>	安装基于 IPv6 的 Astra Trident	false
<code>k8sTimeout</code>	Kubernetes 操作超时	30sec
<code>silenceAutosupport</code>	不要向NetApp发送AutoSupport捆绑包自动	false
<code>autosupportImage</code>	AutoSupport 遥测的容器映像	"netapp/trident-autosupport:24.02"
<code>autosupportProxy</code>	用于发送AutoSupport的代理的地址/端口遥测	"http://proxy.example.com:8888"
<code>uninstall</code>	用于卸载 Astra Trident 的标志	false
<code>logFormat</code>	要使用的 Astra Trident 日志记录格式 [text , json]	"text"
<code>tridentImage</code>	要安装的 Astra Trident 映像	"netapp/trident:24.02"

参数	Description	Default
imageRegistry	内部注册表的路径、格式 <registry FQDN>[:port][/subpath]	"k8s.gcr.io/sig-storage" (Kubernetes 1.19及更高版本) 或 "quay.io/k8s/scsi"
kubeletDir	主机上的 kubelet 目录的路径	"/var/lib/kubelet"
wipeout	要删除以执行完全删除的资源列表 Astra Trident	
imagePullSecrets	从内部注册表中提取映像的机密信息	
imagePullPolicy	设置Trident运算符的映像提取策略。有效值为： Always 以始终提取映像。 IfNotPresent 仅当节点上尚不存在映像时才提取该映像。 Never 从不提取映像。	IfNotPresent
controllerPluginNodeSelector	Pod的其他节点选择器。遵循与相同的格式 pod.spec.nodeSelector。	无默认值；可选
controllerPluginTolerations	覆盖Kubernetes对Pod的容错。遵循与相同的格式 pod.spec.Tolerations。	无默认值；可选
nodePluginNodeSelector	Pod的其他节点选择器。遵循与相同的格式 pod.spec.nodeSelector。	无默认值；可选
nodePluginTolerations	覆盖Kubernetes对Pod的容错。遵循与相同的格式 pod.spec.Tolerations。	无默认值；可选



有关格式化POD参数的详细信息，请参阅 ["将 Pod 分配给节点"](#)。

有关强制断开的详细信息

可对使用强制断开 `ontap-san` 和 `ontap-san-economy` 仅限。在启用强制断开之前，必须在Kubernetes集群上启用非正常节点关闭(NGN)。有关详细信息，请参见 ["Kubnetes：节点非正常关闭"](#)。



由于Astra三端存储依赖于Kubernetes NGN、因此请勿删除 `out-of-service` 运行状况不正常的节点会导致出现问题、直到重新计划所有不可支持的工作负载为止。不负责任地应用或删除该问题可能会危及后端数据保护。

当Kubnetes集群管理员应用了 `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` 此节点和存在污染 `enableForceDetach` 设置为 `true`，Asta Trident将确定节点状态，并：

1. 停止挂载到该节点的卷的后端I/O访问。
2. 将Astra三端节点对象标记为 `dirty` (对于新出版物不安全)。



在节点重新通过资格认定(标记为后)之前、三端技术(Trident)控制器将拒绝新的发布卷请求 dirty)。只有在Asta三端存储能够验证使用已挂载PVC的任何工作负载(即使在集群节点运行状况良好且已准备就绪后)、此工作负载才会被接受 clean (可安全发布新出版物)。

在恢复节点运行状况并删除此污染后、Asta Trident将：

1. 确定并清除节点上陈旧的已发布路径。
2. 如果节点位于中 cleanable 状态(已删除服务中断的部分、并且节点处于状态 Ready 状态)、并且所有过时的已发布路径都是干净的、Asta三端技术将节点重新提交为 clean 并允许向节点发布新的已发布卷。

配置示例

您可以使用中的属性 [\[配置选项\]](#) 定义时 TridentOrchestrator 自定义安装。

基本自定义配置

这是一个基本自定义安装示例。

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


节点选择器

此示例将安装具有节点选择器的Asta Trident。

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

Windows工作节点

此示例将A作用于Windows工作节点上。

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

AKS集群上的受管身份

此示例将安装A作用于在AKS集群上启用受管身份。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
```

AKS集群上的云身份

此示例将在AKS集群上安装用于云身份的Asta Dent。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxxx'
```

EKS集群上的云身份

此示例将在AKS集群上安装用于云身份的Asta Dent。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/astratrident-role'"
```

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。