



管理和监控Trident Trident

NetApp
March 03, 2025

目录

管理和监控Trident	1
升级Trident	1
升级Trident	1
使用操作员升级	2
使用 tridentctl 进行升级	7
使用tridentctrd管理Trident	7
命令和全局标志	7
命令选项和标志	9
插件支持	14
监控Trident	14
概述	14
第 1 步：定义 Prometheus 目标	14
第 2 步：创建 Prometheus ServiceMonitor	14
第 3 步：使用 PromQL 查询 Trident 指标	15
了解Trident AutoSupport遥测	16
禁用Trident指标	17
卸载 Trident	17
确定原始安装方法	17
卸载TRident操作员安装	18
卸载 tridentctl 安装	19

管理和监控Trident

升级Trident

升级Trident

从24.02版开始、Trident按四个月的发布节奏、每年发布三个主要版本。每个新版本都是在先前版本的基础上构建的、并提供了新功能、性能增强功能、错误修复和改进。我们建议您至少每年升级一次、以利用Trident中的新功能。

升级前的注意事项

升级到最新版本的Trident时、请考虑以下事项：

- 在给定的Kubernetes集群中的所有名称区中只应安装一个Trident实例。
- Trident 23.07及更高版本需要v1卷快照、不再支持Alpha或Beta快照。
- 如果您在中创建了适用于Google Cloud的Cloud Volumes Service"CVS 服务类型"，则在从Trident 23.01升级时，必须更新后端配置才能使用 `standardsw`` 或 `zoneredundantstandardsw`` 服务级别。如果无法在后端更新 `serviceLevel``、可能会导致卷失败。有关详细信息、请参见 "CVS服务类型示例"。
- 升级时、请务必在"由Trident使用"中 `StorageClasses`` 提供 `parameter.fsType``。您可以在不中断已有卷的情况下删除和重新创建 `StorageClasses`` 卷。
 - 这是执行的一项 * 要求 * "安全上下文" SAN 卷。
 - [sample输入目录包含https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-basic.yaml.template](https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-basic.yaml.template)等示例[`storage-class-basic.yaml.template`]和链接：<https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-bronze-default.yaml.template>。
 - 有关详细信息，请参见 "已知问题"。

第1步：选择版本

Trident版本遵循基于日期的 `YY.MM`` 命名约定、其中"YY"是一年的最后两位数字、"MM"是月份。DOT版本遵循 `YY.MM.X`` 惯例、其中"X"是修补程序级别。您将根据要从中升级的版本选择要升级到的版本。

- 您可以直接升级到已安装版本的四个版本窗口中的任何目标版本。例如、您可以直接从24.06 (或任何24.06 DOT版本)升级到25.02。
- 如果要从四个版本窗口之外的版本进行升级、请执行多步骤升级。按照您要升级的的升级说明升级到适合四个版本窗口的 "早期版本" 最新版本。例如、如果您运行的是23.01并希望升级到25.02：
 - a. 首次从23.01升级到24.02。
 - b. 然后从24.02升级到25.02。



在OpenShift容器平台上使用TRIDENT操作程序进行升级时、应升级到TRIDENT 21.01.1或更高版本。21.01.0 版发布的 Trident 运算符包含一个已知的问题描述，该 已在 21.01.1 中修复。有关详细信息、请参见 "[GitHub 上的问题描述详细信息](#)"。

第2步：确定原始安装方法

要确定最初安装Trident时使用的版本、请执行以下操作：

1. 使用 `... kubectl get pods -n trident` 检查Pod。
 - 如果没有操作员POD，则使用安装了Trident `tridentctl`。
 - 如果有操作员POD、则Trident是使用Trident操作员手动或使用Helm安装的。
2. 如果有操作员POD、请使用 ``kubectl describe torc`` 确定Trident是否是使用Helm安装的。
 - 如果有Helm标签、则Trident是使用Helm安装的。
 - 如果没有Helm标签、则Trident是使用Trident操作员手动安装的。

第3步：选择升级方法

通常，您应该使用与初始安装相同的方法进行升级，但您可以这样做"[在安装方法之间切换](#)"。升级Trident有两种方法。

- "[使用Trident操作符进行升级](#)"



我们建议您查看 "[了解操作员升级 workflow](#)" 在使用操作员进行升级之前。

*

使用操作员升级

了解操作员升级 workflow

在使用Trident运算符升级Trident之前、您应了解升级期间发生的后台进程。其中包括对支持滚动更新的三项技术控制器、控制器Pod和节点Pod以及节点DemonSet进行的更改。

TRident操作员升级处理

要安装和升级Trident的众多操作之一"[使用啮合式操作员的优势](#)"是自动处理Trident和Kubbernetes对象、而不会中断已挂载的现有卷。通过这种方式，Trident可以在零停机时间或"[滚动更新](#)"的情况下支持升级。尤其是、通过与Kubenetes集群进行通信、可以：

- 删除并重新创建三级控制器部署和节点DemonSet。
- 使用新版本更换TRIDent控制器Pod和TRIDent节点Pod。
 - 如果节点未更新、则不会阻止更新其余节点。
 - 只有运行了三项节点Pod的节点才能挂载卷。



有关Kubnetes集群上Trident架构的详细信息，请参阅"[Trident架构](#)"。

操作员升级 workflow

使用三端修复操作符启动升级时：

1. 三端运算符*:
 - a. 检测当前安装的Trident版本(版本_n_)。
 - b. 更新所有Kubernetes对象、包括CRD、RBAC和三项服务。
 - c. 删除版本为_n_的TRident控制器部署。
 - d. 创建版本为_n+1_的三项控制器部署。
2. *Kubernetes*为_n+1_创建了三项控制器Pod。
3. 三端运算符*:
 - a. 删除_n_的三项目标节点演示集。操作员不会等待节点Pod终止。
 - b. 为_n+1_创建三项目标节点演示。
4. *Kubernetes*会在未运行三端节点Pod _n_的节点上创建三端节点Pod。这样可以确保一个节点上的任何版本的三端存储节点Pod不会超过一个。

使用Trident Operator或Helm升级Trident安装

您可以使用Trident操作员手动或使用Helm升级Trident。您可以从Trident操作员安装升级到另一个Trident操作员安装、也可以从安装升级 `tridentctl` 到Trident操作员版本。在升级Trident操作员安装之前、请查看["选择升级方法"](#)。

升级手动安装

您可以从集群范围的三端技术人员安装升级到另一个集群范围的三端技术人员安装。所有Trident 21.01及更高版本都使用集群范围的运算符。



要从使用命名空间范围的运算符(版本20.07到20.10)安装的Trident升级、请按照Trident的升级说明进行操作["您安装的版本"](#)。

关于此任务

{\f270通过} {\f270 {\f151、} {\f270} {\f270} {\f151、} {\f270} {\f270} {\f151、} {\f270通过} {\f151、} {\f270} {\f270} {\f151、} {\f270}

- 对于运行Kubernetes 1.24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

确保您使用的是正在运行的Kubernetes集群 ["支持的Kubernetes版本"](#)。

步骤

1. 验证Trident版本:

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

2. 删除用于安装当前Trident实例的Trident运算符。例如、如果要从23.07升级、请运行以下命令:

```
kubectl delete -f 23.07.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

3. 如果您使用自定义了初始安装 `TridentOrchestrator` 属性、您可以编辑 `TridentOrchestrator` 用于修改安装参数的对象。其中可能包括为脱机模式指定镜像 `Trident` 和 `CSI` 映像注册表、启用调试日志或指定映像提取密钥所做的更改。
4. 使用适用于您的环境的正确捆绑包 `YAML` 文件安装 `Trident`、其中 `_是` 或 ``bundle_post_1_25.yaml`` 基于您的 `<bundle.yaml>` ``bundle_pre_1_25.yaml`` 版本。例如、如果要安装 `Trident 25.02`、请运行以下命令：

```
kubectl create -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

升级Helm安装

您可以升级 `Trident Helm` 安装。



在将安装了 `Trident` 的 `Kubernetes` 集群从 `1.24` 升级到 `1.25` 或更高版本时、您必须 `helm upgrade`` 先更新 `values.yaml` 以设置为或添加到 ``true`` 命令、``--set excludePodSecurityPolicy=true`` 然后才能升级集群。 ``excludePodSecurityPolicy`

如果您已将 `Kubernetes` 集群从 `1.24` 升级到 `1.25`、但未升级 `Trident Helm`、则 `Helm` 升级将失败。要完成 `Helm` 升级、请作为前提条件执行以下步骤：

1. 从安装 `helm-mapkubeapis` 插件 <https://github.com/helm/helm-mapkubeapis>。
2. 在安装了 `Trident` 的命名空间中对 `Trident` 版本执行演练。此操作将列出要清理的资源。

```
helm mapkubeapis --dry-run trident --namespace trident
```

3. 使用 `Helm` 执行完整运行以执行清理。

```
helm mapkubeapis trident --namespace trident
```

步骤

1. 如果您 "已使用 `Helm` 安装 `Trident`" 使用、则可以使用 `helm upgrade trident netapp-trident/trident-operator --version 100.2502.0` 一步升级。如果您未添加 `Helm repo` 或无法使用它进行升级：
 - a. 从下载最新的 `Trident` 版本 "[GitHub 上的_assets_部分](#)"。
 - b. 使用 `helm upgrade` 命令、其中 `trident-operator-25.02.0.tgz` 反映了要升级到的版本。

```
helm upgrade <name> trident-operator-25.02.0.tgz
```



如果您在初始安装期间设置了自定义选项(例如、为三端映像和CSI映像指定专用、镜像注册表)、请附加 `helm upgrade` 命令 `--set` 为了确保这些选项包含在 `upgrade` 命令中、否则这些值将重置为默认值。

2. 运行 `helm list` 验证图表和应用程序版本均已升级。运行 `tridentctl logs` 查看任何调试消息。

从升级到 `tridentctl` 安装到 **TRident** 操作员

您可以从升级到最新版本的 **Trident** 操作员 `tridentctl` 安装。现有后端和PVC将自动可用。



在切换安装方法之前、请查看 ["在安装方法之间移动"](#)。

步骤

1. 下载最新版本的 **Trident**。

```
# Download the release required [25.02.0]
mkdir 25.02.0
cd 25.02.0
wget
https://github.com/NetApp/trident/releases/download/v25.02.0/trident-
installer-25.02.0.tar.gz
tar -xf trident-installer-25.02.0.tar.gz
cd trident-installer
```

2. 从清单中创建 `tridentorchator` CRD。

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

3. 将集群范围的运算符部署在同一命名空间中。

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

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
```

NAME	READY	STATUS	RESTARTS	AGE
trident-controller-79df798bdc-m79dc	6/6	Running	0	150d
trident-node-linux-xrst8	2/2	Running	0	150d
trident-operator-5574dbbc68-nthjv	1/1	Running	0	1m30s

4. 创建 `TridentOrchestrator` CR以安装Trident。

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

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
```

NAME	READY	STATUS	RESTARTS	AGE
trident-csi-79df798bdc-m79dc	6/6	Running	0	1m
trident-csi-xrst8	2/2	Running	0	1m
trident-operator-5574dbbc68-nthjv	1/1	Running	0	5m41s

5. 确认已将三项功能升级到预期版本。

```
kubectl describe torc trident | grep Message -A 3

Message:          Trident installed
Namespace:        trident
Status:           Installed
Version:          v25.02.0
```


使用 `tridentctl` 进行升级

您可以使用轻松升级现有的Trident安装 `tridentctl`。

关于此任务

卸载和重新安装Trident相当于升级。卸载Trident时、不会删除Trident部署所使用的永久性卷请求(PVC)和永久性卷(PV)。已配置的PV在Trident脱机时仍可用、Trident将在恢复联机后为在此期间创建的任何PVC配置卷。

开始之前

请查看 ["选择升级方法"](#) 升级之前 `tridentctl`。

步骤

1. 在中运行卸载命令 ``tridentctl`` 以删除与Trident关联的所有资源、CRD和相关对象除外。

```
./tridentctl uninstall -n <namespace>
```

2. 重新安装Trident。请参阅 ["使用tridentctl安装Trident"](#)。



请勿中断升级过程。确保安装程序运行完毕。

使用 `tridentctrd` 管理Trident

<https://github.com/NetApp/trident/releases> ["Trident 安装程序包"^] 包含 ``tridentctl`` 命令行实用程序、可用于轻松访问Trident。具有足够Privileges的Kubernetes用户可以使用它来安装Trident或管理包含Trident Pod的命名空间。

命令和全局标志

您可以运行 `tridentctl help` 以获取可用命令的列表 `tridentctl` 或附加 `--help` 用于任何命令的标志、以获取该特定命令的选项和标志列表。

```
tridentctl [command] [--optional-flag]
```

Trident ``tridentctl`` 实用程序支持以下命令和全局标志。

create

将资源添加到Trident。

delete

从Trident中删除一个或多个资源。

get

从Trident获取一个或多个资源。

help

有关任何命令的帮助。

images

打印Trident所需容器映像的表。

import

将现有资源导入到Trident。

install

安装 Trident 。

logs

从Trident打印日志。

send

从Trident发送资源。

卸载

卸载Trident。

update

在Trident中修改资源。

update backend state

暂时暂停后端操作。

upgrade

在Trident中升级资源。

ve版本

打印Trident版本。

全局标志

-d, --debug

调试输出。

-h, --help

帮助 tridentctl。

-k, --kubeconfig string

指定 KUBECONFIG 在本地或从一个Kubernetes集群到另一个集群运行命令的路径。



或者、您也可以导出 KUBECONFIG 变量、用于指向特定的Kubernetes集群和问题描述 tridentctl 命令。

-n, --namespace string

Trident部署的命名空间。

-o, --output string

输出格式。json_yaml_name_wide|ps 之一（默认）。

-s, --server string

Trident REST接口的地址/端口。



可以将 Trident REST 接口配置为仅以 127.0.0.1（对于 IPv4）或 (:::1)（对于 IPv6）侦听和提供服务。

命令选项和标志

创建

使用 `create` 命令将资源添加到Trident。

```
tridentctl create [option]
```

选项

backend: 将后端添加到Trident。

删除

使用 `delete` 命令从Trident中删除一个或多个资源。

```
tridentctl delete [option]
```

选项

backend: 从Trident中删除一个或多个存储后端。

snapshot: 从Trident中删除一个或多个卷快照。

storageclass: 从Trident中删除一个或多个存储类。
volume: 从Trident中删除一个或多个存储卷。

获取

使用 `get` 命令从Trident获取一个或多个资源。

```
tridentctl get [option]
```

选项

backend: 从Trident获取一个或多个存储后端。
snapshot: 从Trident获取一个或多个快照。
storageclass: 从Trident获取一个或多个存储类。
volume: 从Trident获取一个或多个卷。

标志

-h, --help: 卷帮助。
--parentOfSubordinate string: 将查询限制为从源卷。
--subordinateOf string: 将查询限制为卷的下属。

映像

使用 `images` 标志打印Trident所需容器映像的表。

```
tridentctl images [flags]
```

标志

-h, --help: 图像帮助。
-v, --k8s-version string: Kubernetes集群的语义版本。

导入卷

使用 `import volume` 命令将现有卷导入Trident。

```
tridentctl import volume <backendName> <volumeName> [flags]
```

别名

volume, v

标志

-f, --filename string: YAML或JSON PVC文件的路径。
-h, --help: 卷的帮助。
--no-manage: 仅创建PV/PVC。不要假定卷生命周期管理。

安装

使用 `install` 标志安装Trident。

```
tridentctl install [flags]
```

标志

- `--autosupport-image string`: AutoSupport遥测的容器图像(默认为“NetApp / Trident AutoSupport : <current-version>”)。
- `--autosupport-proxy string`: 用于发送AutoSupport遥测的代理的地址/端口。
- `--enable-node-prep`: 尝试在节点上安装所需的软件包。
- `--generate-custom-yaml`: 生成YAML文件而不安装任何内容。
- `-h --help`: 安装帮助。
- `--http-request-timeout`: 覆盖Trident控制器REST API的HTTP请求超时(默认值为1m30s)。
- `--image-registry string`: 内部映像注册表的地址/端口。
- `--k8s-timeout duration`: 所有KubeNet操作的超时(默认值为3m0)。
- `--kubelet-dir string`: kubelet内部状态的主机位置(默认为“/var/lib/kubelet”)。
- `--log-format string`: Trident日志记录格式(文本, json)(默认为“文本”)。
- `--node-prep`: 使Trident能够使Kubernetes集群的节点做好准备, 以便使用指定的数据存储协议管理卷。目前, **iscsi**是唯一支持的值。
- `--pv string`: Trident使用的原有PV的名称确保不存在(默认为“Trident”)。
- `--pvc string`: Trident使用的原有PVC的名称确保不存在(默认为“Trident”)。
- `--silence-autosupport`: 不自动向NetApp发送AutoSupport分发包(默认为true)。
- `--silent`: 在安装期间禁用大多数输出。
- `--trident-image string`: 要安装的Trident映像。
- `--use-custom-yaml`: 使用安装目录中的任何现有YAML文件。
- `--use-ipv6`: 使用IPv6进行Trident通信。

日志

使用 `logs` 标志从Trident打印日志。

```
tridentctl logs [flags]
```

标志

- `-a --archive`: 创建包含所有日志的支持归档文件, 除非另有说明。
- `-h --help`: 日志帮助。
- `-l --log string`: 要显示的Trident日志。Trident |自动|Auto-operator|all之一Trident (默认值为“auto”)。
- `--node string`: 要从中收集节点Pod日志的Kubernetes节点名称。
- `-p --previous`: 获取上一个容器实例(如果存在)的日志。
- `--sidecars`: 获取此容器的日志。

发送

使用 `send` 命令从Trident发送资源。

```
tridentctl send [option]
```

选项

- `autosupport`: 将AutoSupport 归档发送给NetApp。

卸载

使用 `uninstall` 标志卸载Trident。

```
tridentctl uninstall [flags]
```

标志

- h, --help: 卸载帮助。
- silent: 卸载期间禁用大多数输出。

更新

使用 `update` 命令修改 Trident 中的资源。

```
tridentctl update [option]
```

选项

`backend`: 在 Trident 中更新后端。

更新后端状态

使用 `update backend state` 用于暂停或恢复后端操作的命令。

```
tridentctl update backend state <backend-name> [flag]
```

需要考虑的要点

- 如果使用 TridentBackendConfig (tbc) 创建后端、则无法使用文件更新后端 `backend.json`。
- 如果 `userState` 已在 tbc 中设置、则无法使用命令修改 `tridentctl update backend state <backend-name> --user-state suspended/normal`。
- 要在通过 tbc 设置后重新能够通过 `tridentctl` 设置 `userState`、必须从 tbc 中删除此字段。 `userState` 可以使用命令来完成此操作 `kubectl edit tbc`。删除此字段后 `userState`、您可以使用 `tridentctl update backend state` 命令更改 `userState` 后端的。
- 使用 `tridentctl update backend state` 更改 `userState`。您还可以使用或文件更新 `userState` TridentBackendConfig `backend.json`；这会触发后端的完全重新初始化、并且可能会非常耗时。

标志

- h, --help: 后端状态帮助。
- user-state: 设置为 `suspended` 暂停后端操作。设置为 `normal` 恢复后端操作。设置为 `suspended`:

- `AddVolume` 和 `Import Volume` 已暂停。
- `CloneVolume`、`ResizeVolume` `PublishVolume`、`UnPublishVolume`、`CreateSnapshot` `GetSnapshot` `RestoreSnapshot`、`DeleteSnapshot`、`RemoveVolume`、`GetVolumeExternal` `ReconcileNodeAccess` 保持可用。

您也可以使用后端配置文件或中的字段更新后端状态 `userState` TridentBackendConfig `backend.json`。有关详细信息，请参阅 ["用于管理后端的选项"](#) 和 ["使用 kubectl 执行后端管理"](#)。

- 示例: *

JSON

按照以下步骤使用文件更新 `userState backend.json` :

1. 编辑 `backend.json` 文件以包含 `userState` 字段、并将其值设置为"已附加"。
2. 使用命令和更新后的文件的路径更新后端 `tridentctl backend update backend.json` 。

示例: `tridentctl backend update -f /<path to backend JSON file>/backend.json`

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "<redacted>",
  "svm": "nas-svm",
  "backendName": "customBackend",
  "username": "<redacted>",
  "password": "<redacted>",
  "userState": "suspended",
}
```

YAML

您可以在使用命令应用`trident`后对其进行编辑 `kubectl edit <trident-name> -n <namespace>`。以下示例使用选项将后端状态更新为暂停 `userState: suspended` :

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  backendName: customBackend
  storageDriverName: ontap-nas
  managementLIF: <redacted>
  svm: nas-svm
  userState: suspended
  credentials:
    name: backend-tbc-ontap-nas-secret
```

version

使用 `... version` 用于打印版本的标志 `tridentctl` 以及正在运行的Trident服务。

```
tridentctl version [flags]
```

标志

- client: 仅限客户端版本(不需要服务器)。
- h, --help: 版本帮助。

插件支持

tridentctd支持类似于kubectd的插件。如果插件二进制文件名遵循"tridentcts-tld"方案<plugin>、并且二进制文件位于列出了路径环境变量的文件夹中、则tridentctl将检测插件。所有检测到的插件都会在tridentctrd帮助的插件部分中列出。您也可以通过在环境变量TRIDENTCTL_plugin_path中指定plugin文件夹来限制搜索(示例: TRIDENTCTL_PLUGIN_PATH=~/.tridentctl-plugins/)。如果使用了变量、则tridentctl仅在指定文件夹中搜索。

监控Trident

Trident提供了一组Prometheus指标端点、可用于监控Trident性能。

概述

通过Trident提供的指标、您可以执行以下操作:

- 密切关注Trident的运行状况和配置。您可以检查操作的成功程度以及它是否能够按预期与后端进行通信。
- 检查后端使用情况信息,并了解在后端配置的卷数量以及占用的空间量等。
- 维护可用后端配置的卷数量的映射关系。
- 跟踪性能。您可以查看Trident与后端通信并执行操作所需的时间。



默认情况下, Trident 的指标会显示在 /metrics 端点的目标端口`8001`上。安装 Trident 时, 这些指标默认为 * 已启用 *。

您需要的内容

- 安装了Trident的Kubernetes集群。
- 一个 Prometheus 实例。可以是 ["容器化 Prometheus 部署"](#) 或者, 您也可以选择将 Prometheus 作为运行 ["原生应用程序"](#)。

第 1 步: 定义 Prometheus 目标

您应定义Prometheus目标、以收集指标并获取有关Trident管理的后端及其创建的卷等的信息。这 ["博客"](#)说明了如何将Prometheus和Grafana与Trident结合使用来检索指标。本博客介绍了如何以操作员身份在Kubernetes集群中运行Prometheus、以及如何创建ServiceMonitor来获取Trident指标。

第 2 步: 创建 Prometheus ServiceMonitor

要使用 Trident 指标, 您应创建一个 Prometheus ServiceMonitor, 该监控器可监控 trident CSI 服务并侦听 metrics 端口。示例 ServiceMonitor 如下所示:


```

apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - port: metrics
      interval: 15s

```

此ServiceMonitor定义检索服务返回的指标 `trident-csi`、并专门查找 `metrics` 服务的端点。因此、Prometheus现在可以配置为了解Trident的指标。

除了直接从Trident获得的指标之外、kubelet还会通过自己的指标端点公开许多 `kubelet_volume_*` 指标。Kubelet 可以提供有关已连接的卷、Pod 及其处理的其他内部操作的信息。请参阅 ["此处"](#)。

第 3 步：使用 PromQL 查询 Trident 指标

PromQL 非常适合创建返回时间序列或表格数据的表达式。

您可以使用以下 PromQL 查询：

获取 **Trident** 运行状况信息

- 来自**Trident**的**HTTP 2XX**响应的百分比

```

(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100

```

- 通过状态代码来自**Trident**的**REST**响应的百分比

```

(sum (trident_rest_ops_seconds_total_count) by (status_code) / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100

```

- 由**Trident**执行的操作的平均持续时间(以毫秒为单位)

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

获取Trident使用情况信息

- 卷大小 * 平均值 *

```
trident_volume_allocated_bytes/trident_volume_count
```

- * 每个后端配置的卷总空间 *

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

获取单个卷的使用情况



只有在同时收集 kubelet 指标时，才会启用此功能。

- * 每个卷的已用空间百分比 *

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

了解Trident AutoSupport遥测

默认情况下、Trident会每天向NetApp发送Prometheus指标和基本后端信息。

- 要阻止Trident向NetApp发送Prometheus指标和基本后端信息、请在Trident安装期间传递此 `--silence-autosupport` 标志。
- Trident还可以通过按需向NetApp支持发送容器日志 `tridentctl send autosupport`。您需要触发Trident来上传其日志。在提交日志之前，应接受NetApp的<https://www.netapp.com/company/legal/privacy-policy/>["隐私政策"]。
- 除非指定、否则Trident将提取过去24小时的日志。
- 您可以使用标志指定日志保留时间范围 `--since`。例如：`tridentctl send autosupport --since=1h`。此信息通过随Trident一起安装的容器收集和发送 `trident-autosupport`。您可以从获取容器映像 "[Trident AutoSupport](#)"。
- Trident AutoSupport 不会收集或传输个人身份信息（PII）或个人信息。它附带的 "[EULA](#)" 不适用于三端存储容器映像本身。您可以详细了解NetApp对数据安全和信任的承诺 "[此处](#)"。

Trident发送的有效负载示例如下所示：

```

---
items:
- backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
  protocol: file
  config:
    version: 1
    storageDriverName: ontap-nas
    debug: false
    debugTraceFlags:
    disableDelete: false
    serialNumbers:
    - nwkvzfanek_SN
    limitVolumeSize: ''
  state: online
  online: true

```

- AutoSupport 消息将发送到 NetApp 的 AutoSupport 端点。如果使用私有注册表存储容器映像，则可以使用 `-image-regRegistry` 标志。
- 您也可以生成安装 YAML 文件来配置代理 URL。为此，可以使用 `tridentctl install -generate-custom-yaml` 创建 YAML 文件，并在 `trident dedeployment.yaml` 中为 `trident autosupport` 容器添加 `-proxy-url` 参数。

禁用Trident指标

要报告指标，您应生成自定义 YAML（使用 `-generate-custom-yaml` 标志）并对其进行编辑，以删除为 `trident` 主容器调用的 `-metrics` 标志。

卸载 Trident

您应使用与安装Trident相同的方法卸载Trident。

关于此任务

- 如果您需要修复在升级、依赖关系问题或升级失败或不完整后发现的错误，则应卸载Trident，然后按照适用于该版本的特定说明重新安装早期版本"`version`"。这是将_降级_降级到早期版本的唯一建议方法。
- 为了便于升级和重新安装、卸载Trident不会删除Trident创建的CRD或相关对象。如果需要完全删除Trident及其所有数据，请参见"[完全删除Trident和CRD](#)"。

开始之前

如果要停用Kubernetes集群、则必须在卸载之前删除使用Trident创建的卷的所有应用程序。这样可以确保在删除之前、在Kubernetes节点上未取消对这些PVC的审核。

确定原始安装方法

您应使用与安装Trident相同的方法来卸载它。卸载之前、请验证最初安装Trident时使用的版本。

1. 使用 `... kubectl get pods -n trident` 检查Pod。
 - 如果没有操作员POD，则使用安装了Trident `tridentctl`。
 - 如果有操作员POD、则Trident是使用Trident操作员手动或使用Helm安装的。
2. 如果有操作员POD、请使用 ``kubectl describe tproc trident`` 确定Trident是否是使用Helm安装的。
 - 如果有Helm标签、则Trident是使用Helm安装的。
 - 如果没有Helm标签、则Trident是使用Trident操作员手动安装的。

卸载TRident操作员安装

您可以手动卸载或使用Helm卸载TRYDent操作员安装。

卸载手动安装

如果您使用操作员安装了Trident、则可以通过执行以下操作之一将其卸载：

1. 编辑 **TridentOrchestrator CR**并设置卸载标志*：

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

当 `uninstall` 标志设置为 `true`、Trident操作符将卸载Trident、但不会删除Trident Orchestrator本身。如果要重新安装 Trident ，应清理 Trident Orchestrator 并创建新的 Trident 。

2. 删除 **TridentOrchestrator**:删除用于部署Trident的CR后 `TridentOrchestrator`，您将指示操作员卸载Trident。操作员将处理Trident部署和守护进程的删除过程、并继续删除此部署和守护进程 `TridentOrchestrator`、同时删除其在安装过程中创建的Trident Pod。

```
kubectl delete -f deploy/<bundle.yaml> -n <namespace>
```

卸载Helm安装

如果您使用Helm安装了Trident，则可以使用将其卸载 `helm uninstall`。

```
#List the Helm release corresponding to the Trident install.
helm ls -n trident
NAME                NAMESPACE      REVISION      UPDATED
STATUS              CHART           APP VERSION
trident             trident         1             2021-04-20
00:26:42.417764794 +0000 UTC deployed    trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
helm uninstall trident -n trident
release "trident" uninstalled
```

卸载 tridentctl 安装

使用 `uninstall` 中的命令 `tridentctl` 删除与Trident关联的所有资源(CRD和相关对象除外):

```
./tridentctl uninstall -n <namespace>
```

版权信息

版权所有 © 2025 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。