# 管理和监控 Trident

Trident

NetApp
July 01, 2026

# 目录

# 管理和监控 Trident

# 升级 Trident

## 升级 Trident

从 24.02 版本开始，Trident 遵循四个月的发布节奏，每个日历年发布三个主要版本。每个新版本都建立在先前版本的基础上，并提供新功能、性能增强、错误修复和改进。我们建议您每年至少升级一次，以利用 Trident 中的新功能。

### 升级前的注意事项

升级到最新版 Trident 时，请考虑以下事项：

- 在给定的 Kubernetes 集群中的所有命名空间中应仅安装一个 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。
- 升级时，请务必提供 Trident 使用的 `parameter.fsType`中的 `StorageClasses`。您可以删除和重新创建 `StorageClasses`而不会中断先前存在的卷。
  - 这是强制执行 "安全上下文" SAN 卷的要求。
  - sample input 目录包含示例，例如 `storage-class-basic.yaml.templ` 和 `storage-class-bronze-default.yaml`。
  - 有关详细信息，请参阅 "已知问题"。

### 步骤 1：选择一个版本

Trident 版本遵循基于日期的 `YY.MM`命名约定，其中"YY"是年份的最后两位数字，"MM"是月份。点版本遵循 `YY.MM.X`约定，其中"X"是补丁级别。您将根据要升级的版本选择要升级到的版本。

- 您可以直接升级到安装版本的四个版本窗口内的任何目标版本。例如，您可以直接从 24.06（或任何 24.06 dot 版本）升级到 25.06。
- 如果您要从四版本窗口之外的版本进行升级，请执行多步骤升级。使用您要从"早期版本"升级的升级说明升级到适合四发布窗口的最新版本。例如，如果您运行的是 23.07 并希望升级到 25.06：
  a. 首次从 23.07 升级到 24.06。
  b. 然后从 24.06 升级到 25.06。

> 在 OpenShift Container Platform 上使用 Trident 操作员升级时，应升级到 Trident 21.01.1 或更高版本。随 21.01.0 发布的 Trident 操作员包含一个已在 21.01.1 中修复的已知问题。有关更多详细信息，请参阅 "在 GitHub 上的问题详细信息"。

### 步骤 2：确定原始安装方法

要确定您最初用于安装 Trident 的版本：

1. 使用 `kubectl get pods -n trident` 检查 pod。

- 如果没有操作员舱，则说明 Trident 是使用 `tridentctl` 安装的。
- 如果有 operator pod，Trident 是使用 Trident operator 手动或使用 Helm 安装的。

2. 如果有操作员 pod，请使用 `kubectl describe torc` 来确定是否使用 Helm 安装了 Trident。
   - 如果有 Helm 标签，则说明 Trident 是使用 Helm 安装的。
   - 如果没有 Helm 标签，则使用 Trident 操作员手动安装 Trident。

步骤 **3**：选择一种升级方法

一般来说，您应该使用与初始安装相同的方法进行升级，但您可以"在安装方法之间移动"。升级 Trident 有两种选择。

- "使用 Trident 操作员进行升级"

我们建议您在使用 operator 升级之前查看"了解 operator 升级工作流程"。

*

## 使用 operator 进行升级

了解 **operator** 升级工作流程

在使用 Trident 操作员升级 Trident 之前，您应该了解升级过程中发生的后台过程。这包括对 Trident 控制器、控制器 Pod 和节点 Pod 以及启用滚动更新的节点 DaemonSet 的更改。

**Trident** 操作员升级处理

安装和升级 Trident 的众多"使用 Trident 运算符的好处"之一是自动处理 Trident 和 Kubernetes 对象，而不会中断现有的已挂载卷。通过这种方式，Trident 可以支持零停机时间升级，或"滚动更新"。特别是，Trident 操作符与 Kubernetes 集群通信以：

- 删除并重新创建 Trident Controller 部署和节点 DaemonSet。
- 将 Trident Controller Pod 和 Trident Node Pod 替换为新版本。
  - 如果节点未更新，则不会阻止更新剩余节点。
  - 只有运行 Trident Node Pod 的节点才能挂载卷。

有关 Kubernetes 集群上的 Trident 架构的更多信息，请参见 "Trident 架构"。

**Operator** 升级工作流

使用 Trident 操作员启动升级时：

1. **Trident** 操作员：
   a. 检测当前安装的 Trident 版本（version *n*）。
   b. 更新所有 Kubernetes 对象，包括 CRD、RBAC 和 Trident SVC。

c. 删除版本 *n* 的 Trident Controller 部署。

d. 为版本 *n+1* 创建 Trident Controller 部署。

2. **Kubernetes** 为 *n+1* 创建 Trident Controller Pod。

3. **Trident** 操作员：

a. 删除 *n* 的 Trident 节点 DaemonSet。操作员不等待节点 Pod 终止。

b. 为 *n+1* 创建 Trident 节点守护进程集。

4. **Kubernetes** 在不运行 Trident Node Pod *n* 的节点上创建 Trident Node Pod。这可确保一个节点上不会有多于一个任何版本的 Trident Node Pod。

使用 **Trident** 操作员或 **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，或从 `tridentctl`安装升级到 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

您可以从集群范围内的 Trident 操作员安装升级到另一个集群范围内的 Trident 操作员安装。所有 Trident 版本都使用集群范围的操作符。

> 要从使用命名空间范围运算符（版本 20.07 至 20.10）安装的 Trident 进行升级，请使用 "您安装的版本" 版本的 Trident 升级说明。

关于此任务

Trident 提供了一个捆绑文件，可用于安装操作员并为您的 Kubernetes 版本创建关联的对象。

- 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

请确保使用的是正在运行 "受支持的 Kubernetes 版本" 的 Kubernetes 集群。

步骤

1. 验证您的 Trident 版本：

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

2. 使用要升级到的版本（例如 25.06）的注册表和图像路径以及正确的密码更新 `operator.yaml`、`tridentorchestrator_cr.yaml` 和 `post_1_25_bundle.yaml`。

3. 删除用于安装当前 Trident 实例的 Trident 操作员。例如，如果要从 25.02 升级，请运行以下命令：

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

4. 如果使用 `TridentOrchestrator` 属性自定义初始安装，则可以编辑 `TridentOrchestrator` 对象以修改安装参数。这可能包括为离线模式指定镜像的 Trident 和 CSI 映像注册表、启用调试日志或指定映像拉取密钥所做的更改。
5. 使用适合您环境的正确捆绑包 YAML 文件安装 Trident，其中 *<bundle.yaml>* 是 `bundle_pre_1_25.yaml`或 `bundle_post_1_25.yaml`，具体取决于您的 Kubernetes 版本。例如，如果要安装 Trident 25.06.0，请运行以下命令：

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. 编辑 Trident torc 以包括图像 25.06.0。

#### 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，必须更新 values.yaml 以将 `excludePodSecurityPolicy`设置为 `true`或将 `--set excludePodSecurityPolicy=true`添加到 `helm upgrade`命令中，然后才能升级集群。

如果您已经将 Kubernetes 集群从 1.24 升级到 1.25，而没有升级 Trident helm，则 helm 升级将失败。要完成 helm 升级，请执行以下步骤作为先决条件：

1. 从 https://github.com/helm/helm-mapkubeapis安装 helm-mapkubeapis 插件。
2. 在安装 Trident 的命名空间中对 Trident 版本执行试运行。这列出了将要清除的资源。

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

3. 使用 helm 执行完整运行以进行清理。

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

#### 步骤

1. 如果您"使用 Helm 安装了 Trident"，您可以使用 `helm upgrade trident netapp-trident/trident-operator --version 100.2602.0`一步升级。如果您没有添加 Helm 存储库或无法使用它进行升级：
   a. 从 "GitHub 上的 *Assets* 部分" 下载最新的 Trident 版本。
   b. 请使用 `helm upgrade` 命令，其中 `trident-operator-26.02.0.tgz` 反映要升级到的版本。

```
helm upgrade <name> trident-operator-26.02.0.tgz
```

如果您在初始安装期间设置了自定义选项（例如为 Trident 和 CSI 镜像指定私有、镜像仓库），请使用 `helm upgrade`命令并附加 `--set`，以确保这些选项包含在升级命令中，否则这些值将重置为默认值。

2. 运行 `helm list` 以确认图表和应用版本都已升级。运行 `tridentctl logs` 以查看所有调试消息。

从 `tridentctl` 安装升级到 **Trident** 操作员

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

在切换安装方法之前，请查看 "在安装方法之间切换"。

步骤

1. 下载最新的 Trident 版本。

```
# Download the release required [26.02.0]
mkdir 26.02.0
cd 26.02.0
wget
https://github.com/NetApp/trident/releases/download/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

2. 从清单中创建 `tridentorchestrator` CRD。

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

3. 在同一命名空间中部署集群作用域运算符。

```
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 Trident 已升级到预期版本。

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

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v26.02.0
```

## 使用 tridentctl 升级

### 您可以使用 `tridentctl` 轻松升级现有 Trident 安装。

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 部署使用的 Persistent Volume Claim (PVC) 和 Persistent Volume (PV)。已配置的 PV 将在 Trident 脱机时保持可用，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

在使用 `tridentctl`进行升级之前，请先"选择升级方法"审核。

步骤

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

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

2. 重新安装 Trident。请参见 "使用 tridentctl 安装 Trident"。

> 不要中断升级过程。确保安装程序运行完成。

# 使用 tridentctl 管理 Trident

`https://github.com/NetApp/trident/releases["Trident 安装包"^]` 包括 `tridentctl` 命令行实用程序，用于提供对 `Trident` 的简单访问。拥有足够权限的 `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 `tridentctl`的可用命令列表，或将 `--help`标志附加到任何命令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。

命令

**create**
向 Trident 添加资源。

**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 需要的容器镜像表。

**import**
将现有资源导入到 Trident。

**install**
安装 Trident。

**logs**
打印 Trident 的日志。

**send**
从 Trident 发送资源。

**uninstall**
卸载 Trident。

**update**
修改 Trident 中的资源。

**update backend state**
暂时挂起后端操作。

**upgrade**
升级 Trident 中的资源。

**version**
打印 Trident 的版本。

全局标记

**-d, --debug**

调试输出。

**-h, --help**

`tridentctl` 的帮助。

**-k, --kubeconfig string**

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

或者，您可以导出 `KUBECONFIG` 变量以指向特定的 Kubernetes 集群，并向该集群发出 `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 接口配置为仅在 127.0.0.1（用于 IPv4）或 [::1]（用于 IPv6）下侦听和服务。

## 命令选项和标志

### create

使用 `create` 命令可将资源添加到 Trident。

```
tridentctl create [option]
```

选项

`backend`：向 Trident 添加后端。

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。

标记

`-h, --help`: 卷的帮助。
`--parentOfSubordinate string`: 将查询限制为从属源卷。
`--subordinateOf string`: 将查询限制为卷的下属。

镜像

使用 `images`标志打印 Trident 需要的容器镜像表。

```
tridentctl images [flags]
```

标记

`-h, --help`: 图片帮助。
`-v, --k8s-version string`: Kubernetes 集群的语义版本。

导入卷

使用 `import volume` 命令可将现有卷导入 Trident。

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

别名

`volume, v`

标记

`-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。不要假设卷生命周期管理。

安装

使用 `install` 标志安装 Trident。

```
tridentctl install [flags]
```

标记

`--autosupport-image string`：Autosupport Telemetry 的容器映像（默认为 "netapp/trident autosupport:<current-version>"）。
`--autosupport-proxy string`：用于发送 Autosupport Telemetry 的代理的地址/端口。
`--enable-node-prep`：尝试在节点上安装所需的程序包。
`--generate-custom-yaml`：在不安装任何内容的情况下生成 YAML 文件。
`-h， --help`：安装帮助。
`--http-request-timeout`：覆盖 Trident 控制器的 REST API 的 HTTP 请求超时（默认为 1m30s）。
`--image-registry string`：内部映像注册表的地址/端口。
`--k8s-timeout duration`：所有 Kubernetes 操作的超时（默认为 3m0s）。
`--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（text、json）（默认为 "text"）。
`--node-prep`：使 Trident 能够准备 Kubernetes 集群的节点，以使用指定的数据存储协议管理卷。当前，**`iscsi`** 是唯一受支持的值。从 **OpenShift 4.19** 开始，此功能支持的最低 **Trident** 版本为 **25.06.1**。
`--pv string`：Trident 使用的旧版 PV 的名称，确保其不存在（默认为 "trident"）。
`--pvc string`：Trident 使用的旧版 PVC 的名称，确保其不存在（默认为 "trident"）。
`--silence-autosupport`：不要自动将 autosupport 捆绑包发送到 NetApp（默认为 true）。
`--silent`：安装期间禁用大多数输出。
`--trident-image string`：要安装的 Trident 映像。
`--k8s-api-qps`：Kubernetes API 请求的每秒查询数 (QPS) 限制（默认为 100；可选）。
`--use-custom-yaml`：使用安装目录中存在的任何现有 YAML 文件。
`--use-ipv6`：使用 IPv6 进行 Trident 通信。

### logs

使用 `logs` 标志打印 Trident 的日志。

```
tridentctl logs [flags]
```

标记

`-a, --archive`: 除非另有指定，否则创建包含所有日志的支持存档。
`-h, --help`: 日志帮助。
`-l, --log string`: 要显示的 Trident 日志。trident|auto|trident-operator|all 之一（默认为 "auto"）。
`--node string`: 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的日志（如果存在）。
`--sidecars`: 获取 sidecar 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 存档发送到 NetApp。

卸载

使用 `uninstall` 标志卸载 Trident。

```
tridentctl uninstall [flags]
```

标记

`-h, --help`：有关卸载的帮助。
`--silent`：在卸载期间禁用大多数输出。

更新

使用 `update` 命令可修改 Trident 中的资源。

```
tridentctl update [option]
```

选项

`backend`：更新 Trident 中的后端。

更新后端状态

使用 `update backend state` 命令可挂起或恢复后端操作。

```
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 `backend.json` 文件更新该后端。
- 如果 `userState`已在 tbc 中设置，则无法使用 `tridentctl update backend state <backend-name> --user-state suspended/normal`命令对其进行修改。
- 要在通过 tbc 设置 `userState`后重新获得通过 `tridentctl` 设置的能力，必须从 `tbc` 中删除 `userState`字段。这可以使用 `kubectl edit tbc`命令来完成。删除 `userState`字段后，您可以使用 `tridentctl update backend state`命令来更改后端的 `userState`。
- 使用 `tridentctl update backend state`来更改 `userState`。你也可以通过 `TridentBackendConfig`或 `backend.json`文件来更新 `userState`；这会触发后端的完全重新初始化，并且可能会耗费较多时间。

  标记

  `-h, --help`：后端状态的帮助。
  `--user-state`：设置为 `suspended` 以暂停后端操作。设置为 `normal` 以恢复后端操作。当设置为 `suspended` 时：

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

您还可以使用后端配置文件中的 `userState`字段来更新后端状态 `TridentBackendConfig`或 `backend.json`。有关更多信息，请参阅"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 `userState` 文件更新 `backend.json`：

1. 编辑 `backend.json` 文件以包含值设置为 'suspended' 的 `userState` 字段。
2. 使用 `tridentctl update backend` 命令和更新的 `backend.json` 文件路径更新后端。

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

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

**YAML**

您可以使用 `kubectl edit <tbc-name> -n <namespace>` 命令在应用 tbc 后对其进行编辑。以下示例使用 `userState: suspended` 选项将后端状态更新为挂起：

```
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` 标志打印 `tridentctl` 的版本和正在运行的 Trident 服务。

```
tridentctl version [flags]
```

标记

`--client`：仅客户端版本（无需服务器）。
`-h, --help`：版本帮助。

### 插件支持

Tridentctl 支持类似于 kubectl 的插件。如果插件二进制文件名遵循"tridentctl-<plugin>"方案，并且二进制文件位于 PATH 环境变量列出的文件夹中，则 Tridentctl 会检测插件。所有检测到的插件都列在 tridentctl 帮助的插件部分中。或者，您也可以通过在环境变量 TRIDENTCTL_PLUGIN_PATH 中指定插件文件夹来限制搜索（示例：`TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`）。如果使用该变量，tridentctl 仅在指定的文件夹中搜索。

# 监控 Trident

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

## 概述

Trident 提供的指标使您能够执行以下操作：

- 密切关注 Trident 的健康状况和配置。您可以检查操作的成功程度，以及它是否可以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上配置了多少卷以及占用的空间量等。
- 维护可用后端上配置的卷数量的映射。
- 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 的指标在目标端口 `8001`的 `/metrics`端点上公开。安装 Trident 时，这些指标*默认启用*。您也可以配置在端口 `8444`上通过 HTTPS 使用 Trident 指标。

您需要什么

- 已安装 Trident 的 Kubernetes 集群。
- Prometheus 实例。这可以是 "容器化 Prometheus 部署"，也可以选择将 Prometheus 作为 "本机应用程序" 运行。

## 步骤 1：定义 Prometheus 目标

您应该定义一个 Prometheus 目标，以收集指标并获取有关 Trident 管理的后端、其创建的卷等信息。请参阅"Prometheus Operator 文档"。

## 步骤 2：创建 Prometheus ServiceMonitor

要使用 Trident 指标，您应该创建一个 Prometheus ServiceMonitor，用于监视 `trident-csi`服务并在 `metrics`端口上侦听。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 `trident-csi` 服务返回的指标，并特别查找服务的 `metrics` 端点。因此，Prometheus 现在被配置为理解 Trident 的指标。

除了可直接从 Trident 获得的指标外，kubelet 还通过自己的 `kubelet_volume_*`指标端点公开了许多指标。Kubelet 可以提供有关附加的卷及其处理的 Pod 和其他内部操作的信息。请参阅 "此处"。

通过 **HTTPS** 使用 **Trident** 指标

要通过 HTTPS（端口 8444）使用 Trident 指标，必须修改 ServiceMonitor 定义以包含 TLS 配置。您还需要将 `trident-csi` 密钥从 `trident` 命名空间复制到运行 Prometheus 的命名空间。您可以使用以下命令执行此操作：

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

HTTPS 指标的 ServiceMonitor 示例如下所示：

```yaml
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:
    - interval: 15s
      path: /metrics
      port: https-metrics
      scheme: https
      tlsConfig:
        ca:
          secret:
            key: caCert
            name: trident-csi
        cert:
          secret:
            key: clientCert
            name: trident-csi
        keySecret:
          key: clientKey
          name: trident-csi
        serverName: trident-csi
```

Trident 支持在所有安装方法中使用 HTTPS 指标：tridentctl、Helm chart 和 Operator：

- 如果使用 `tridentctl install` 命令，则可以传递 `--https-metrics` 标志以启用 HTTPS 指标。
- 如果使用 Helm 图表，可以设置 `httpsMetrics` 参数以启用 HTTPS 指标。
- 如果您正在使用 YAML 文件，您可以在 `trident-deployment.yaml` 文件中的 `trident-main` 容器添加 `--https_metrics` 标志。

## 步骤 3：使用 PromQL 查询 Trident 指标

PromQL 适用于创建返回时间序列或表格数据的表达式。

以下是可以使用的一些 PromQL 查询：

获取 **Trident** 运行状况信息

- 来自 **Trident** 的 **HTTP 2XX** 响应百分比

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

- 通过状态代码从 **Trident** 获得的 **REST** 响应百分比

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

- **Trident** 执行操作的平均持续时间（毫秒）

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

获取 **Trident** 使用信息

- 平均卷大小

```
trident_volume_allocated_bytes/trident_volume_count
```

- 每个后端配置的总卷空间

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

获取单个卷使用量

只有当还收集了 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 将 Prometheus 指标和基本后端信息按每日节奏发送到 NetApp。

- 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 `--silence-autosupport` 标志。
- Trident 还可以通过 `tridentctl send autosupport` 按需将容器日志发送到 NetApp Support。您需要触发 Trident 上传其日志。在提交日志之前，您应接受 NetApp 的 "隐私政策"。
- 除非另有说明，否则 Trident 会从过去 24 小时内获取日志。
- 您可以使用 `--since`标志指定日志保留时间范围。例如： `tridentctl send autosupport --since=1h`。此信息通过与 Trident 一起安装的 `trident-autosupport`容器收集和发送。您可以在 "Trident AutoSupport"获取容器镜像。
- Trident AutoSupport 不会收集或传输个人身份信息 (PII) 或个人信息。它附带 "最终用户许可协议"，不适用于 Trident 容器图像本身。您可以了解有关 NetApp 对数据安全和信任承诺的更多信息 "此处"。

Trident 发送的示例负载如下所示：

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

- AutoSupport 消息将发送到 NetApp 的 AutoSupport 端点。如果要使用专用注册表存储容器图像，则可以使用 `--image-registry` 标志。
- 您还可以通过生成安装 YAML 文件来配置代理 URL。这可以通过使用 `tridentctl install --generate-custom-yaml`来创建 YAML 文件，并为 `trident-autosupport`容器在 `trident-deployment.yaml`中添加 `--proxy-url` 参数来实现。

## 禁用 Trident 指标

要禁用指标上报，您应生成自定义 YAML 文件（使用 `--generate-custom-yaml`参数），并编辑它们以移除 `--metrics`参数，使其不再被 `trident-main`容器调用。

# 卸载 Trident

您应该使用与安装 Trident 相同的方法来卸载 Trident。

关于此任务

- 如果您需要修复升级后发现的错误、依赖问题或未成功或未完成的升级，则应卸载 Trident 并使用特定说明重新安装早期版本"version"。这是_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 不会删除由 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，请参阅 "完全移除 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所有使用 Trident 创建的卷的应用程序。这可确保 PVC 在删除之前未在 Kubernetes 节点上发布。

## 确定原始安装方法

您应该使用与安装 Trident 时相同的方法来卸载 Trident。在卸载之前，请验证您最初用于安装 Trident 的版本。

1. 使用 `kubectl get pods -n trident` 检查 pod。
   - 如果没有操作员舱，则说明 Trident 是使用 `tridentctl` 安装的。
   - 如果有 operator pod，Trident 是使用 Trident operator 手动或使用 Helm 安装的。
2. 如果有操作员 pod，请使用 `kubectl describe tproc trident` 来确定是否使用 Helm 安装了 Trident。
   - 如果有 Helm 标签，则说明 Trident 是使用 Helm 安装的。
   - 如果没有 Helm 标签，则使用 Trident 操作员手动安装 Trident。

## 卸载 Trident 操作员安装

您可以手动或使用 Helm 卸载 Trident 操作员安装。

卸载手动安装

如果您使用操作员安装了 Trident，可以通过执行以下操作之一来卸载它：

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`时，Trident 操作员卸载 Trident，但不删除 TridentOrchestrator 自身。如果要再次安装 Trident，您应该清理 TridentOrchestrator 并创建一个新的。

2. 删除 **`TridentOrchestrator`**：通过删除用于部署 Trident 的 `TridentOrchestrator` CR，您指示操作员卸载 Trident。操作员处理 `TridentOrchestrator`的删除并继续删除 Trident 部署和守护程序集，删除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 ，可以使用 `helm uninstall`将其卸载。

```
#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` 命令在 `tridentctl` 中删除与 Trident 关联的所有资源，但 CRD 和相关对象除外：

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